MATLAB 7.0 从入门到精通

求是科技　编著

人 民 邮 电 出 版 社

图书在版编目（CIP）数据

MATLAB 7.0 **从入门到精通／求是科技编著．—北京：人民邮电出版社，**2006.3

ISBN 7-115-14327-7

Ⅰ．M… **Ⅱ．求**… **Ⅲ．计算机辅助计算－软件包，**MATLAB 7.0 **Ⅳ．**TP391.75

中国版本图书馆 CIP **数据核字**（2006）**第** 007936 **号**

内 容 提 要

本书对 MATLAB 7.0 进行了详细的介绍，力求做到细致全面。全书共分为 15 章。前 5 章是有关 MATLAB 的基础知识，包括 MATLAB 的安装、卸载及系统功能的简述，MATLAB 的数学运算和数据可视化工具以及 MATLAB 的编程等内容。第 6 章～第 9 章是 MATLAB 的高级应用部分，分别介绍了 MATLAB 7.0 的数据分析和处理功能，符号计算功能，Simulink 仿真环境和文件 I/O 操作。第 10 章～第 12 章和第 14 章介绍了信号处理工具箱、图像处理工具箱和编译工具箱以及应用广泛的高级图形设计。第 13 章、第 15 章介绍了 MATLAB 的外围功能，包括与 Word、Excel 的混合使用和 MATLAB 的应用程序接口。附录部分列出了常用命令和函数，Simulink 的库模块和应用程序接口函数库。

本书叙述详细，深入浅出，又有丰富的例程，适合使用 MATLAB 的本科生、研究生和教师以及广大科研工作人员作为参考用书。

MATLAB 7.0 从入门到精通

♦ 编　　著　求是科技
　责任编辑　张立科

♦ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
　邮编　100061　　电子函件　315@ptpress.com.cn
　网址　http://www.ptpress.com.cn
　北京密云春雷印刷厂印刷
　新华书店总店北京发行所经销

♦ 开本：787×1092　1/16
　印张：35.25
　字数：855 千字　　　　2006 年 3 月第 1 版
　印数：1 – 5 000 册　　　2006 年 3 月北京第 1 次印刷

ISBN 7-115-14327-7/TP・5185

定价：54.00 元（附光盘）

读者服务热线：(010)67132692　印装质量热线：(010)67129223

前　言

MATLAB 是 Mathworks 公司于 1984 年推出的一套高性能的数值计算和可视化软件，它集数值分析、矩阵运算、信号处理和图形显示于一体，可方便地应用于数学计算、算法开发、数据采集、系统建模和仿真、数据分析和可视化、科学和工程绘图、应用软件开发等方面。MATLAB 之所以能够被广泛应用，是因为它将科研工作者从乏味的 Fortran、C 编程中解放出来，使他们真正把精力放在科研和设计的核心问题上，从而大大提高了工作效率。在 MATLAB 环境中描述问题及编制求解问题的程序时，用户可以按照符合人们的科学思维方式和数学表达习惯的语言形式来书写程序。

MATLAB 这个词代表"矩阵实验室"（matrix laboratory），它是以著名的线性代数软件包 LINPACK 和特征值计算软件包 EISPACK 中的子程序为基础，发展而成的一种开放型程序设计语言。在它的发展过程中，许多优秀的工程师为它的完善做出了卓越的贡献，使其从一个简单的矩阵分析软件逐渐发展成为一个具有极高通用性的、带有众多实用工具的运算操作平台。工具箱是 MATLAB 函数的子程序库，每一个工具箱都是为某一类学科专业和应用而定制的，主要包括信号处理、控制系统、神经网络、图像处理、模糊逻辑、小波分析和系统仿真等方面的应用。借助于这些现有的工具，科研人员可以直观、方便地进行分析、计算及设计工作，从而大大节省了时间。

本书主要是从使用方面对 MATLAB 7.0 进行了详细的介绍，大部分内容是通用的，也包括一些专业性较强的章节，读者可以根据需要取舍。前五章介绍 MATLAB 的基础知识，包括 MATLAB 的安装卸载及其系统功能的阐述，MATLAB 的数学运算和数据可视化工具以及 MATLAB 的编程等内容。接下来四章是 MATLAB 的高级应用部分，分别介绍了 MATLAB 7.0 的数据分析和处理功能，符号计算功能，Simulink 仿真环境和文件 I/O 操作。本书还对一些应用广泛的工具箱（信号处理工具箱、图像处理工具箱和编译工具箱）单独进行了讲述，最后介绍了高级图形设计的相关知识和 MATLAB 与 Word、Excel 的混合使用以及 MATLAB 的应用程序接口。

为读者查阅方便，附录部分列出了 MATLAB 的常用命令和函数，Simulink 的库模块和应用程序接口函数库。

本书叙述详细，深入浅出，又有丰富的例程，适合学习或使用 MATLAB 的本科生、研究生和教师以及广大科研工作人员作为参考用书。除了篇幅和专业所限，有一些工具箱没有介绍之外，MATLAB 系统的各个部分的功能都有比较详细的阐述，只要认真阅读，一定会有收获。

由于时间仓促和作者的水平有限，书中难免有缺点和错误，敬请广大读者批评指正。欢迎广大读者访问求是科技网站 http://www.cs-book.com，提出你的宝贵意见和建议，同时也欢迎感兴趣的读者访问求是论坛做进一步交流。

编者

2006 年 2 月

目　录

第 1 章　MATLAB 概述

MATLAB 是一种高效的工程计算语言，它将计算、可视化和编程等功能集于一个易于使用的环境。在 MATLAB 环境中描述问题及编制求解问题的程序时，用户可以按照符合人们科学思维的方式和数学表达习惯的语言形式来书写程序。其典型应用主要包括以下几个方面：

- 数学计算；
- 算法开发；
- 数据采集；
- 系统建模和仿真；
- 数据分析和可视化；
- 科学和工程绘图；
- 应用软件开发（包括用户界面）。

MATLAB 是一个交互式系统（写程序与执行命令同步），其基本的数据元素是没有维数限制的阵列。这使得用户可以解决许多工程技术上的问题，特别是那些包含了矩阵和向量的公式的计算。采用 MATLAB 编制解决上述问题的程序比采用只支持标量和非交互式的编程语言（如 C 语言和 Fortan 语言）更加方便。

MATLAB 这个词代表“矩阵实验室”(matrix laboratory)，它是以线性代数软件包 LINPACK 和特征值计算软件包 EISPACK 中的子程序为基础发展起来的一种开放型程序设计语言。20 世纪 80 年代初期，Cleve Moler 和 John Little 采用 C 语言改写了 MATLAB 的内核，不久他们便成立了 Mathworks 软件开发公司，并将 MATLAB 正式推向市场。历经十几年的发展和竞争，MATLAB 成为国际认可的最优化的科技应用软件。在大学里，它是用于初等和高等数学、自然科学和工程学的标准教学工具；在工业界，它是一个高效的研究、开发和分析的工具。随着科技的发展，许多优秀的工程师不断地对 MATLAB 进行了完善，使其从一个简单的矩阵分析软件逐渐发展成为一个具有极高通用性，并带有众多实用工具的运算操作平台。

MATLAB 的一个重要特色就是它有一套程序扩展系统和一组称之为工具箱（toolboxes）的特殊应用子程序。工具箱是 MATLAB 函数的子程序库，每一个工具箱都是为某一类学科专业和应用而定制的，主要包括信号处理、控制系统、神经网络、模糊逻辑、小波分析和系统仿真等方面的应用。

MATLAB 系统由以下 5 个主要部分组成，下面具体进行介绍。

- 开发环境：由一系列工具组成。这些工具方便用户使用 MATLAB 的函数和文件，其中许多工具采用的是图形用户界面。包括 MATLAB 桌面和命令窗口、历史命令窗口、编辑器和调试器、路径搜索和用于浏览帮助、工作空间、文件的浏览器。
- MATLAB 数学函数库：这是一个包含大量计算算法的集合，这些函数包括从最简单最基本的函数（如加、正弦等）到诸如矩阵的特征向量、快速傅立叶变换等较复杂的函数。
- MATLAB 语言：这是一个高级的矩阵/阵列语言，它包含控制语句、函数、数据结构、输入输出和面向对象的编程特点。用户可以在命令窗口中将输入语句与执行命令同步，

也可以先编写好一个较大的复杂的应用程序（M 文件）后再一起运行。

- 图形处理：用 MATLAB 可以将向量和矩阵用图形表现出来，并且可以对图形进行标注和打印。高层次的作图包括二维和三维数据可视化、图像处理、动画和表达式作图，低层次的作图包括定制图形的显示和为用户的 MATLAB 应用程序建立的图形用户界面。
- MATLAB 应用程序接口（API）：这是一个库，它允许用户编写可以和 MATLAB 进行交互的 C 或 Fortran 语言程序。

1.1 MATLAB 7.0 简介

MATLAB 软件从 1984 年推出的第 1 个版本到目前发布的第 14 个版本 MATLAB 7.0（Release14），有了较大的改进和增补，增加了许多新功能和更为有效的处理方法。

1．开发环境

- 新的用户界面环境和开发环境，使用户更方便地控制多个文件和图形窗口，用户可以按照自己的习惯来定制桌面环境，还可以为常用的命令定义快捷键；
- 功能更强的数组编辑器和工作空间浏览器，用户可更方便地浏览、编辑和图形化变量；
- 提供的 M-Lint 代码分析器，可以方便用户修改代码以取得更好的性能和可维护性；
- 更强大的编辑器，用户可以选择执行 M 文件中的部分内容等。

2．编程

- 支持函数嵌套、有条件中断点；
- 可以用匿名函数来定义单行函数等。

3．数值处理

- 整数算法，方便用户处理更大的整数；
- 单精度算法、线性代数、FFT 和滤波，方便用户处理更大的单精度数据；
- Linsolve 函数，用户可以通过定义系数矩阵更快地求解线性系统；
- ODE 求解泛函数，操作隐式差分等式和求解多点式边界值问题。

4．图形化

- 新的绘图界面窗口，用户可以不必通过输入 M 函数代码而直接在绘图界面窗口中交互式地创建并编辑图形；
- 用户可以直接从图形窗口中生成 M 代码文件，使得用户可以多次重复地执行用户自定义的作图；
- 更强大的图形标注和处理功能，包括对象对齐、连接注释和数据点的箭头等；
- 数据探测工具，用户可以在图形窗口中方便地查询图形上某一点的坐标值；
- 功能更强大的图形句柄等。

5．图形用户界面

- 面板和分组按钮使得用户可以对用户界面的控件进行分组；

- 用户可以直接在 GUIDE 中访问 ActiveX 控件。

6．文件 I/O 和外部应用程序接口

- 新的文件 I/O 函数支持用户可以读更大的文本文件，并且可以向 Excel 和 HDF5 文件中写入内容；
- 支持压缩格式的 MAT 文件，使得用户可以使用较少的磁盘空间保存大量的数据，而且速度更快；
- 可以使用 Javaaddpath 函数来动态添加、删除或重载 Java 类，而不必重启 MATLAB 7.0；
- 支持 COM 用户接口、服务器事件和 Visual Basic 脚本；
- 可以基于简单的对象访问协议（SOAP）来访问网页服务器；
- 提供 FTP 对象用于连接 FTP 服务器，实现对异地文件的处理；
- 支持 Unicode 国际字符集标准，使得 MAT 文件中的字符数据可以在不同语言之间共享。

1.2　MATLAB 7.0 的安装、退出与卸载

1．对硬件和软件的要求

MATLAB 7.0（Release 14）可以安装到下列操作平台上：

- Windows 2000（Service Pack 3 或 4）；
- Windows NT 4.0（Service Pack 5 或 6a）；
- Windows XP；
- Linux ix86 2.4.x, glibc 2.2.5；
- Sun Solaris 2.8 和 2.9；
- HPUX 11.0 和 11.1；
- Mac OS X 10.3.2。

无论在单机还是网络环境，MATLAB 都可发挥其卓越的性能。若单纯地使用 MATLAB 语言进行编程，而不必连接外部语言的程序，则 MATLAB 语言编写出来的程序可以不做任何修改直接移植到其他机型上去使用。MATLAB 7.0 对 PC 机系统的要求如表 1-1 所示。

表 1-1　MATLAB 7.0 对系统的要求

操作平台	Windows XP、Windows 2000 (Service Pack 3 or 4)、Windows NT 4.0 (Service Pack 5 or 6a)
处理器	Pentium III、4、Xeon、 Pentium M、AMD Athlon、Athlon XP、Athlon MP
存储空间	345 MB (仅包括帮助系统的 MATLAB)
内存	256 MB (最小)，512 MB (推荐)
显卡	16-bit、24-bit 或 32-bit 兼容 OpenGL 的图形适配卡 (强烈推荐)
软件	图形加速卡、打印机、声卡 为了运行 MATLAB Notebook、MATLAB Builder for Excel、Excel Link、Database Toolbox、 and MATLAB Web Server，还必须安装 Office 2000 或 Office XP

续表

编译器	为了创建自己的 MEX 文件，则至少需要下列产品之一：DEC Visual Fortran 5.0、Microsoft Visual C/C++4.2 或 5.0、Borland C/C++5.0 或 5.02 Watcom 10.6 或 11

2．安装过程

随着 MATLAB 版本的更新，安装也越来越简便。对于 MATLAB 7.0，用户只要按照安装界面的提示逐步进行即可。下面介绍在 Windows 系统下的安装过程。

- Step 1：安装前的准备

准备好安装密码（PLP）；退出正在运行的其他版本的 MATLAB；确保系统满足安装的要求；获得系统用户的许可权；最好不要在安装过程中运行病毒扫描程序，因为这样会降低安装速度。

- Step 2：开始安装

插入 MATLAB 7.0 光盘到光驱，在 MATLAB 7.0 目录下直接运行"Setup.exe"程序，显示初始化画面，随之显示准备安装的进度条。一般情况下系统会自动搜索到 autorun 文件并进入安装界面。随后出现"Welcome to the MathWorks Installer"对话框，如图 1-1 所示。

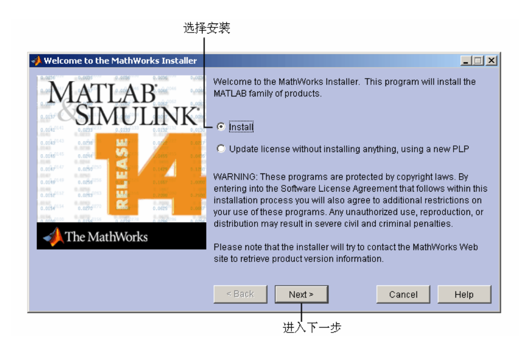


图 1-1 "Welcome to the MathWorks Installer"对话框

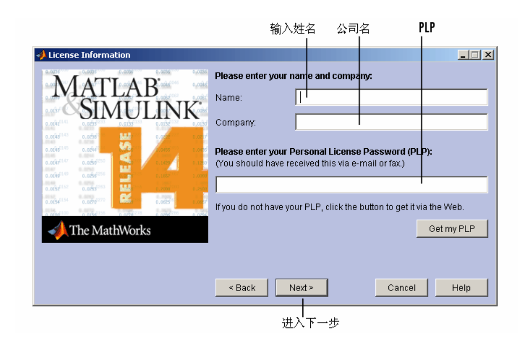


图 1-2 "License Information"对话框

- Step 3：输入用户信息

选择图 1-1 中的"Next"继续安装，出现"License Information"对话框，如图 1-2 所示。在对话框界面的相应位置输入相应的内容，然后单击"下一步"。

- Step 4：浏览如图 1-3 所示的软件许可协议（License Agreement）

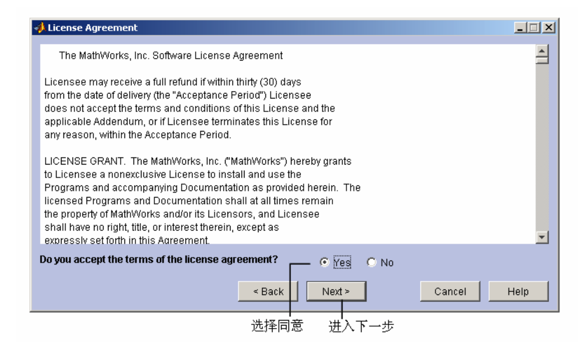


图 1-3 "License Agreement"对话框

• Step 5：选择安装类型

用户可以在如图 1-4 所示的“InstallactionType”对话框中选择安装类型，典型（Typical）或自定义（Custom）。

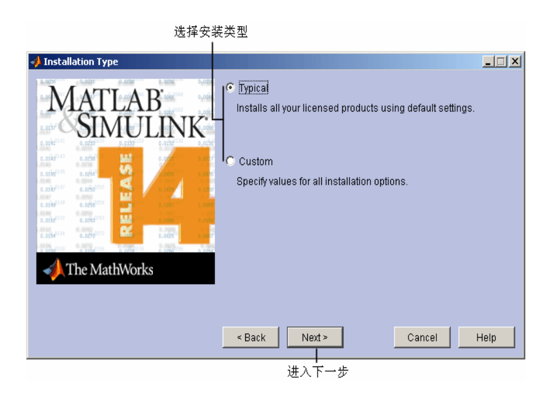


图 1-4　“InstallActionType”对话框

这里选择的是典型安装类型，这样可以简化安装过程，不过安装后会有如下的限制：

① 拥有个人许可证；

② 不能自由地有选择性地安装所需要的产品；

③ 不需要访问安装选项，例如设置某些文件的访问权限等。

如果选择自定义安装类型，可以允许用户选择所要安装的产品，并设定哪些需要访问安装选项。为了保证使用所有产品的各项功能并简化安装过程，一般选择典型安装类型，然后选择进入下一步。

• Step 6：定义安装的目录和所要安装的产品

如果用户选择的是典型安装类型，则只需要设置安装目录；如果选择的是自定义安装类型，则除了设置安装目录外还要选择所需安装的产品。图 1-5 所示为典型安装类型下定义安装目录的界面，图 1-6 是自定义安装类型的设置界面。本例以典型安装类型为例介绍其安装方法。

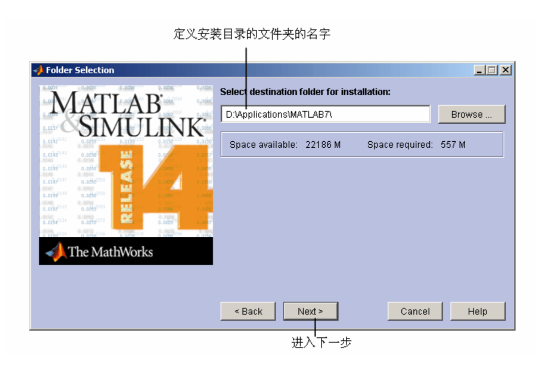


图 1-5　典型安装类型下定义安装目录

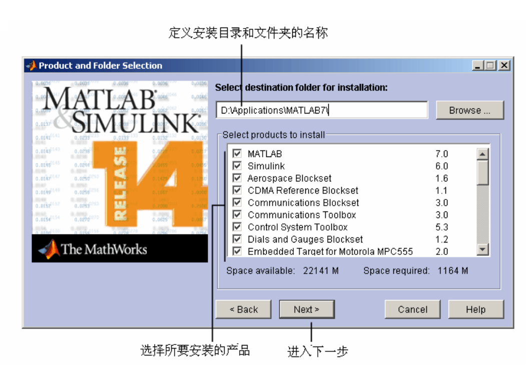


图 1-6　自定义安装类型下定义安装目录和选择产品

• Step 7：确认前面设置的安装目录

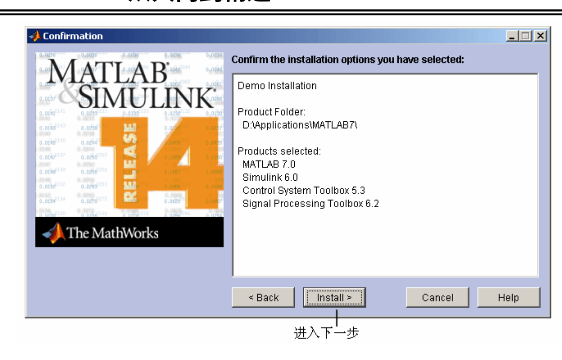


图 1-7　确认安装设置

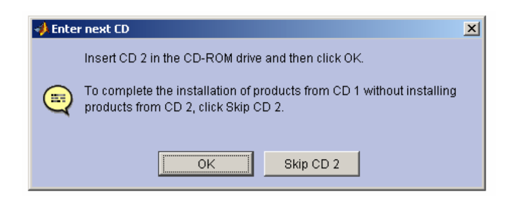


图 1-8　换安装光盘

单击“Install”安装按钮就可以进行安装，并显示一个用以表示当前安装进度的对话框。如果用户所要安装的产品不在当前的光盘上，则会弹出如图 1-8 所示的提示对话框。用户只要按照提示插入另一张光盘即可继续安装。

- Step 8：阅读产品配置报告

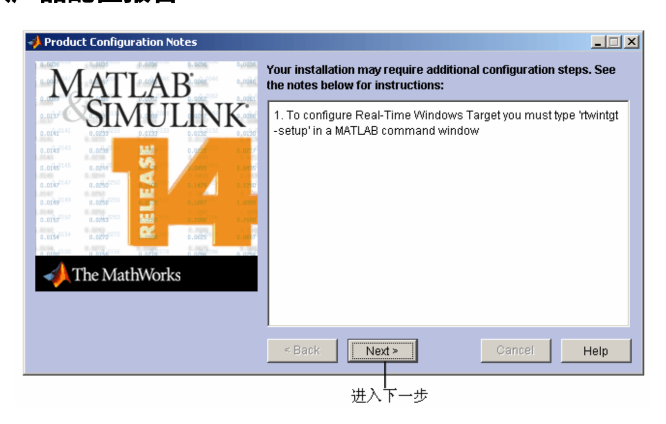


图 1-9　阅览安装情况

- Step 9：完成安装

当安装完毕时，会自动弹出 Setup Complete 对话框。在该对话框中，用户可以选择在退出安装后自动启动 MATLAB 7.0，如果用户不希望退出安装后直接启动它，可以取消选择框。单击“Finish”按钮结束安装。



图 1-10　结束安装

安装完成后，用户可以进行以下工作。

（1）运行 MATLAB 7.0

有 3 种运行 MATLAB 7.0 的方式。用鼠标左键双击桌面的 MATLAB 图标；单击 Start 按钮，选择 Programs，然后在打开的如图 1-11 所示的菜单中选择 MATLAB 7.0；使用 Windows 浏览器打开 MATLAB 7.0 的顶层安装目录，双击快捷运行图标。



图 1-11　通过开始菜单运行 MATLAB 7.0

（2）设置初始目录

在默认情况下，当用户开始运行 MATLAB 7.0 后，默认的初始目录是"$MATLAB\work"。其中"$MATLAB"是指 MATLAB 7.0 的安装路径，work 目录是用于存放修改过的和用户创建的 M-文件，当用户卸载 MATLAB 7.0 后，该目录依然被保存。

用户还可以修改初始的当前目录，用鼠标右键单击桌面上的 MATLAB 7.0 快捷图标，选择 Properties 选项，在"起始位置"的文本区定义自己的初始目录，如图 1-12 所示。



图 1-12　设置初始工作目录

（3）建立 MATLAB 7.0 环境选项

每次在启动 MATLAB 7.0 时，用户可以自定义所要执行的任务，包括欢迎消息、默认定义或任何 MATLAB 表达式。将所有这些任务写在"$MATLAB\toolbox\local"目录下的名称为 startup.m 的文件中，用户每一次启动 MATLAB 7.0 时，系统都会自动执行该文件中的语句。

（4）配置产品

在前面介绍的已安装的某些产品还需要附加配置才能正常使用，在表 1-2 中列出了这些产品以及对它们进行配置的命令，命令的详细使用方法请用户查阅帮助。

表 1-2　配置产品

产品	配置命令
MATLAB Notebook	notebook -setup
Real-Time Windows Target	rtwintgt -setup

（5）查询有关 MATLAB 7.0 的信息

表 1-3 列出了查询 MATLAB 7.0 相应信息的方法。

表 1-3　　查询 MATLAB 7.0 信息

任务	描述
获得对 MATLAB 7.0 总的认识和功能	阅读 MATLAB 7.0 Getting Started 文件
查询所安装版本的新特性	阅读 Release Notes 文件
启动某一产品或访问产品的示例程序	使用桌面的 Start 按钮
获得 MATLAB 7.0 某一特性的相关信息	选择 MATLAB 7.0 主界面的 Help 菜单的相应选项获得相关的 HTML 格式的帮助文件
查询不能在文件中看到的帮助主题	访问 MathWorks 网址（www.mathworks.com）

3．退出 MATLAB 7.0

前面已经介绍了启动 MATLAB 7.0 的方法，下面介绍几种退出 MATLAB 7.0 的方法。

- 在 MATLAB 7.0 命令窗口的“File”菜单下选择“Exit MATLAB”选项；
- 快捷键“Ctrl+q”;
- 在命令窗口输入“quit”;
- 在命令窗口输入“exit”;
- 用鼠标单击 MATLAB 7.0 命令窗口右上角的“ ☒ ”按钮；
- 用鼠标双击 MATLAB 7.0 命令窗口左上角的图标。

4．卸载 MATLAB 7.0

当用户需要卸载 MATLAB 7.0 时，单击系统开始菜单，选择“程序→MATLAB 7.0→R14 Uninstaller”，在弹出的如图 1-13 所示的对话框中选择所要卸载的产品，然后单击“Uninstall”按钮，开始执行卸载命令。

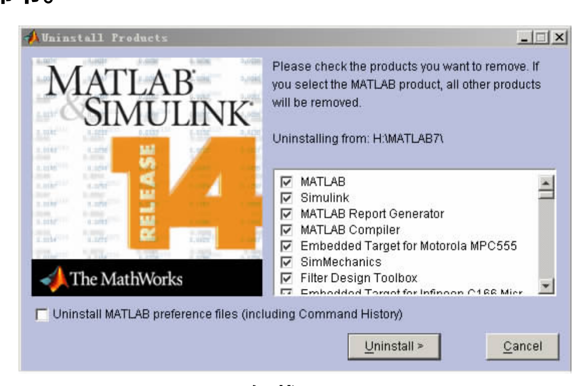


图 1-13　卸载 MATLAB 7.0

1.3　MATLAB 7.0 的目录结构

当用户成功安装 MATLAB 7.0 后，在用户所选择的安装目录下将包含如表 1-4 所示的文件夹目录。

表 1-4　　MATLAB 7.0 的目录结构

文件夹	描述
\BIN\WIN32	MATLAB 7.0 系统中可执行的相关文件
\DEMOS	MATLAB 7.0 示例程序
\EXTERN	创建 MATLAB 7.0 的外部程序接口的工具
\HELP	帮助系统
\JA	MATLAB 7.0 国际化文件
\JAVA	MATLAB 7.0 的 Java 支持程序
\NOTEBOOK	Notebook 是用来实现 MATLAB 数学工作环境与 Word 字处理环境信息交互的软件，是一个兼备数学计算、图形显示和文字处理能力的集成环境
\SYS	MATLAB 7.0 所需要的工具和操作系统库
\TOOLBOX	MATLAB 7.0 的各种工具箱
\UNINSTALL	MATLAB 7.0 的卸载程序
\WORK	默认的当前目录
RTW	Real-Time Workshop 软件包
SIMULINK	Simulink 软件包，用于动态系统的建模、仿真和分析
STATEFLOW	Stateflow 软件包，用于状态机设计的功能强大的图形化开发和设计工具
License.txt	该文件为软件许可协议的内容

1.4　MATLAB 7.0 的工作环境

本节通过介绍 MATLAB 7.0 工作环境界面，使读者初步掌握 MATLAB 7.0 软件的基本操作方法。

MATLAB 7.0 的工作界面主要由菜单、工具栏、当前工作目录窗口、工作空间管理窗口、历史命令窗口和命令窗口组成，如图 1-14 所示。

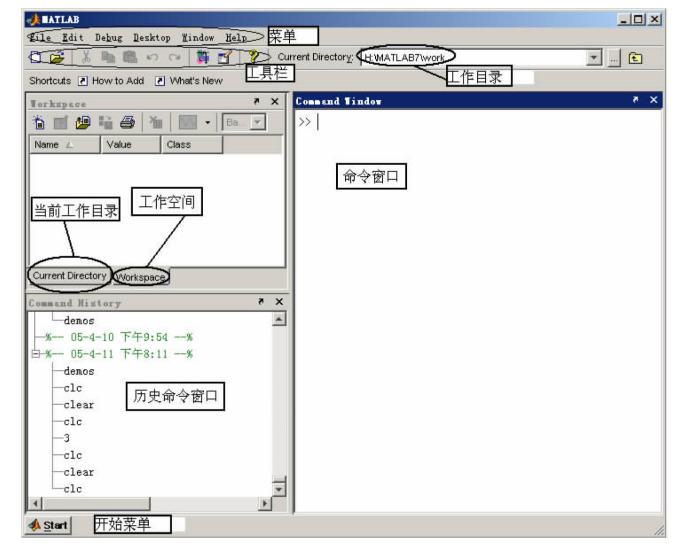


图 1-14　MATLAB 7.0 工作环境

1．菜单和工具栏

MATLAB 7.0 的菜单和工具栏界面与 Windows 程序的界面类似，用户只要稍加实践就可掌握其功能和使用方法。菜单的内容会随着在命令窗口中执行不同命令而作出相应改变。这里只简单介绍默认情况下的菜单和工具栏。

【File】菜单

- Import Data：用于向工作空间导入数据；
- Save Workspace As：将工作空间的变量存储在某一文件中；
- Set path：打开搜索路径设置对话框；
- Preferences：打开环境设置对话框（如图 1-15 所示）。

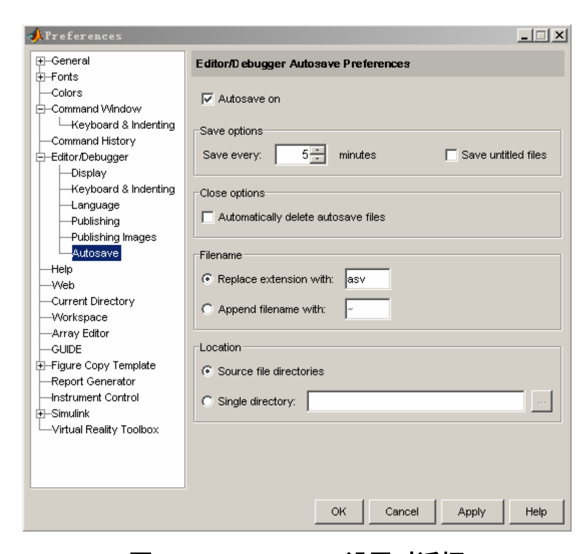


图 1-15　Preferences 设置对话框

【Edit】菜单

主要用于复制、粘贴等操作，与一般的 Windows 程序的类似，在此不作详细介绍。

【Debug】菜单

用于设置程序的调试。

【Desktop】菜单

用于设置主窗口中需要打开的窗口。

【Window】菜单

列出当前所有打开的窗口。

【Help】菜单

用于选择打开不同的帮助系统。

当用户单击“Current Directory”窗口时，使得该窗口成为当前窗口，那么会增加一个如图 1-16 所示的菜单【View】，用于设置如何显示当前目录下的文件。

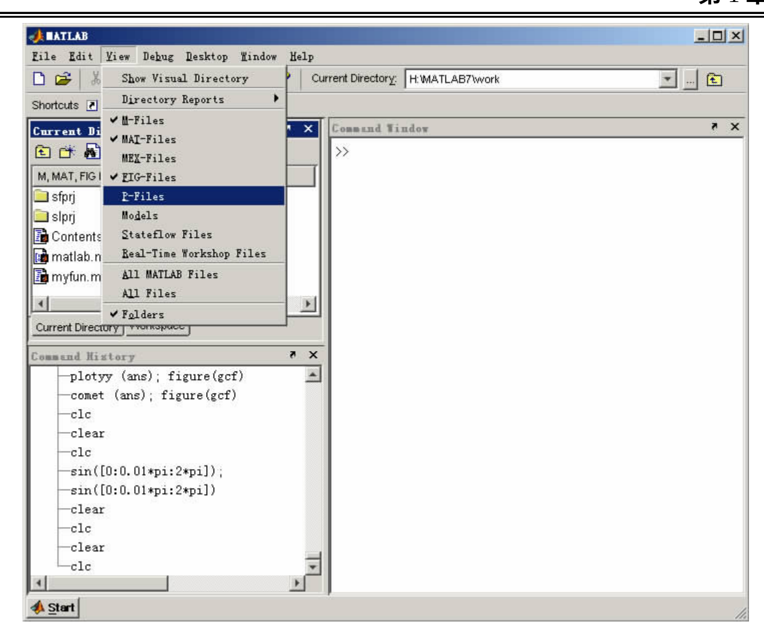


图 1-16　View 菜单

当用户单击“Workspace”窗口时，使得该窗口成为当前窗口，那么会增加如图 1-17 所示的菜单【View】和【Graphics】。菜单【View】用于设置如何在工作空间管理窗口中显示变量，菜单【Graphics】用于打开绘图的工具，用户可以使用这些工具来绘制变量。

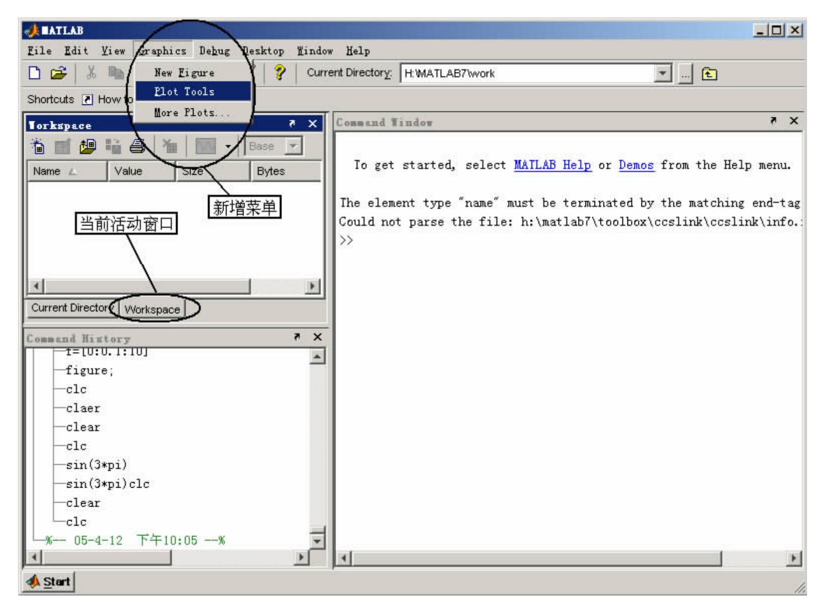


图 1-17　Graphics 菜单

下面介绍“工具栏”中部分按钮的功能。

：打开 Simulink 主窗口。

：打开用户界面设计窗口。

：打开帮助系统。

Current Directory: H:\MATLAB7\work ：设置当前目录。

单击主窗口左下脚的【Start】开始按钮，可以直接打开各种 MATLAB 7.0 工具，如图 1-18 所示。

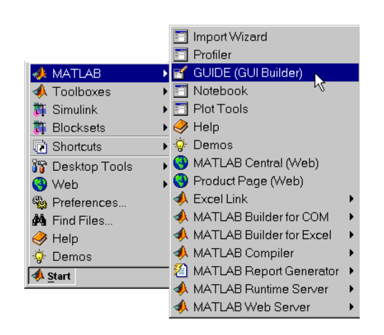


图 1-18　开始按钮

2．命令窗口

MATLAB 7.0 的命令窗口如图 1-19 所示，其中“ >> ”为运算提示符，表示 MATLAB 处于准备状态。当在提示符后输入一段程序或一段运算式后按【Enter】键，MATLAB 会给出计算结果，并再次进入准备状态（所得结果将被保存在工作空间管理窗口中）。

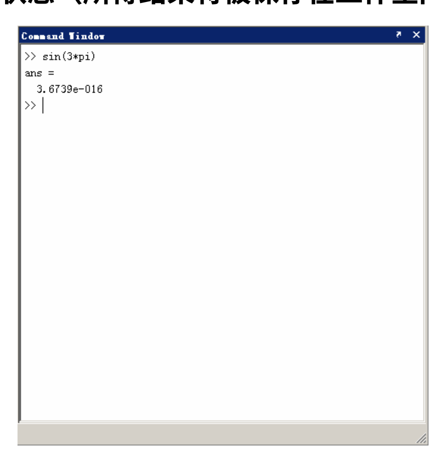


图 1-19　命令窗口

单击命令窗口右上角的按钮，可以使命令窗口脱离主窗口而成为一个独立的窗口，如图 1-20 所示。

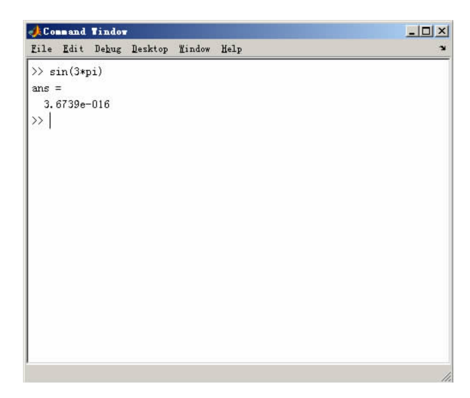


图 1-20　独立的命令窗口

在该窗口中选中某一表达式，然后单击鼠标右键，弹出如图 1-21 所示的上下文菜单，通过不同的选项可以对选中的表达式进行相应的操作。

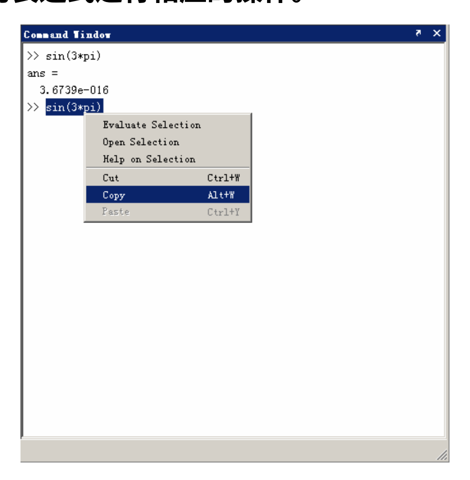


图 1-21　命令窗口的上下文菜单

3．历史命令窗口

该窗口主要用于记录所有执行过的命令，在默认设置下，该窗口会保留自安装后所有使用过的命令的历史记录，并标明使用时间。同时，用户可以通过用鼠标双击某一历史命令来重新执行该命令。与命令窗口类似，该窗口也可以成为一个独立的窗口。

在该窗口中选中，然后单击鼠标右键，弹出如图 1-22 所示的上下文菜单。通过上下文菜单，用户可以删除或粘贴历史记录；也可为选中的表达式或命令创建一个 M-文件；还可为某一句或某一段表达式或命令创建快捷按钮，具体方法见下面的示例。

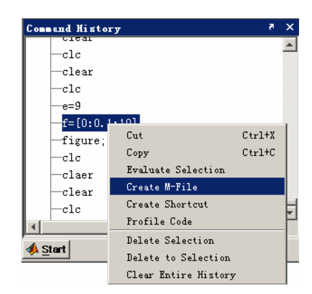


图 1-22　历史命令窗口的上下文菜单

- 选择图 1-22 中的 “ Create Shortcut ” 菜单项，弹出如图 1-23 所示的 “ 快捷键设置 ” 对话框。



图 1-23　“ 快捷键设置 ” 对话框

- 按照图 1-23 进行快捷键的设置，然后单击“Save”按钮，注意观察工具栏 Shortcuts 栏的变化，如图 1-24 所示。

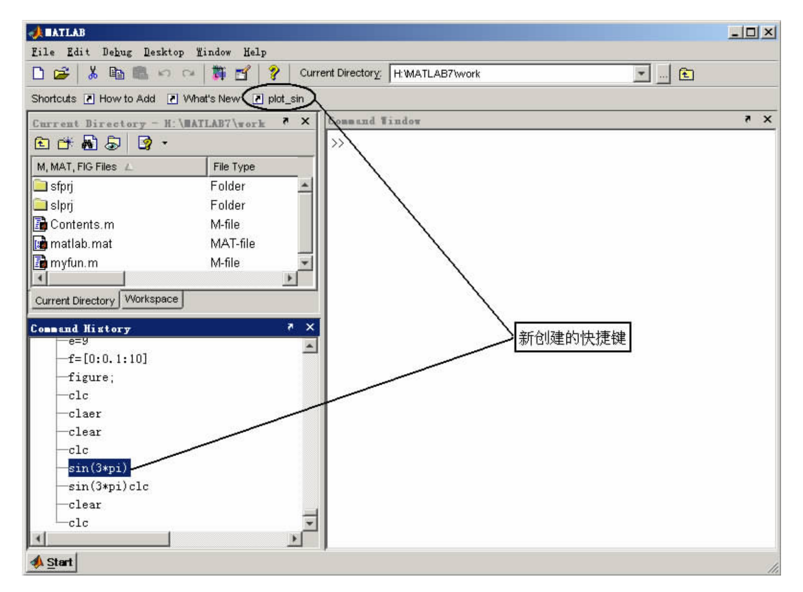


图 1-24　创建新的快捷键

- 用鼠标左键单击新加入的快捷按钮，命令窗口中会显示相应命令执行的结果。

```
ans =
   3.6739e-016
>>
```

用户还可以直接按住鼠标左键不放，将所选中的历史命令直接拖到 Shortcuts 栏中，这样也可为所选命令创建快捷键。

4．当前工作目录窗口

在目录窗口中可显示或改变当前目录，还可以显示当前目录下的文件，以及搜索功能。与命令窗口类似，该窗口也可以成为一个独立的窗口，如图 1-25 所示。

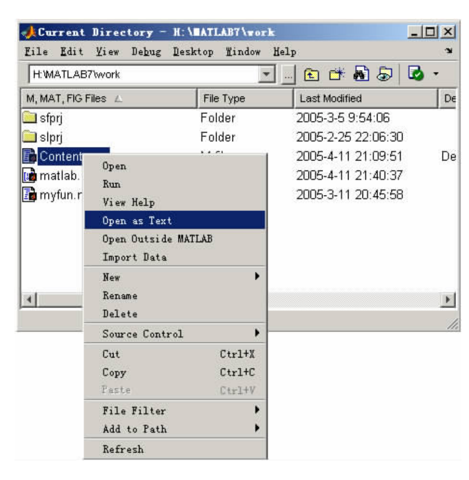


图 1-25　当前目录窗口

下面介绍“当前工作目录窗口”中部分按钮的功能。

H:\MATLAB7\work：显示并改变当前目录；

：进入所显示目录的上一级目录；

：在当前目录中创建一个新的子目录；

：在当前目录中查找一个文件；

：在图 1-25 中选中该按钮后，当前目录中的文件即以类的形式显示，如图 1-26 所示。

：单击该按钮后即可生成一个当前目录中的 M-文件的报告文件，如图 1-27 所示。如果选择该栏下“Contents Report”选项可生成不同的报告文件，如图 1-28 所示。

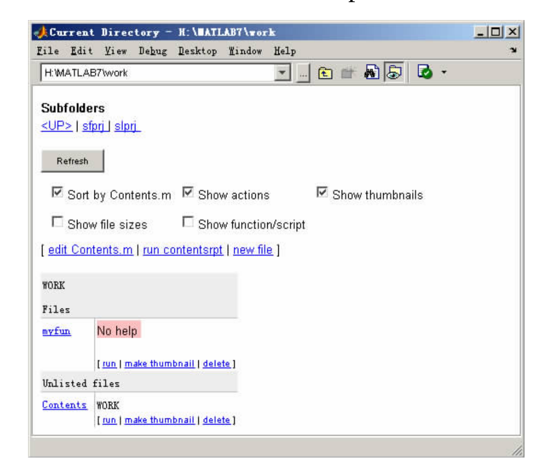


图 1-26　以类的形式显示当前目录中的文件

图 1-27　当前目录中 M-文件的报告文件

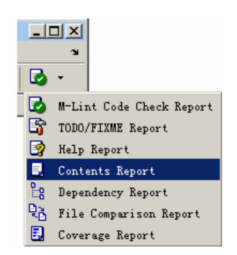


图 1-28　生成不同的报告文件

5．工作空间管理窗口

在工作空间管理窗口中将显示目前内存中所有的 MATLAB 变量的变量名、数据结构、字节数以及类型等信息，不同的变量类型分别对应不同的变量名图标，如图 1-29 所示。

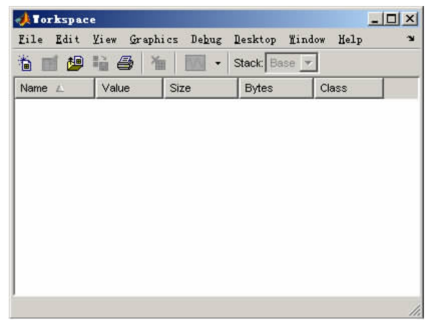


图 1-29　工作空间管理窗口

下面介绍“工作空间管理窗口”中部分按钮的功能。

：向工作空间添加新的变量；

：打开在工作空间中选中的变量；

：向工作空间中导入数据文件；

：保存工作空间中的变量；

：删除工作空间中的变量；

：绘制工作空间中的变量，可以用不同的绘制命令来绘制变量，如图 1-30 所示。

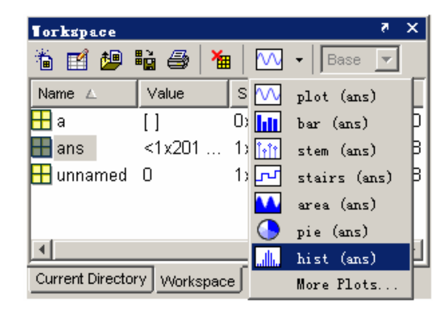


图 1-30　不同的绘制变量命令

1.5　MATLAB 7.0 的通用命令

通用命令是 MATLAB 7.0 中经常使用的一组命令，这些命令可以用来管理目录、命令、函数、变量、工作空间、文件和窗口。为了更好地使用 MATLAB 7.0，用户需要熟练掌握和理解这些命令。下面对这些命令进行介绍。

1．常用命令

常用命令的功能如表 1-5 所示。

表 1-5　常用命令

命令	命令说明	命令	命令说明
cd	显示或改变当前工作目录	load	加载指定文件的变量
dir	显示当前目录或指定目录下的文件	diary	日志文件命令
clc	清除工作窗中的所有显示内容	!	调用 DOS 命令
home	将光标移至命令窗口的最左上角	exit	退出 MATLAB 7.0
clf	清除图形窗口	quit	退出 MATLAB 7.0
type	显示文件内容	pack	收集内存碎片
clear	清理内存变量	hold	图形保持开关
echo	工作窗信息显示开关	path	显示搜索目录
disp	显示变量或文字内容	save	保存内存变量到指定文件

2．输入内容的编辑

在 MATLAB 7.0 命令窗口中，为了便于对输入的内容进行编辑，MATLAB 7.0 提供了一

些控制光标位置和进行简单编辑的一些常用编辑键和组合键，掌握这些可以在输入命令的过程中起到事半功倍的效果。表 1-6 列出了一些常用键盘按键及其作用。

表 1-6　　命令行中的键盘按键

键盘按键	说明	键盘按键	说明
↑	Ctrl+p，调用上一行	home	Ctrl+a，光标置于当前行开头
↓	Ctrl+n，调用下一行	end	Ctrl+e，光标置于当前行末尾
←	Ctrl+b，光标左移一个字符	esc	Ctrl+u，清除当前输入行
→	Ctrl+f，光标右移一个字符	del	Ctrl+d，删除光标处的字符
Ctrl+←	Ctrl+l，光标左移一个单词	backspace	Ctrl+h，删除光标前的字符
Ctrl+→	Ctrl+r，光标右移一个单词	Alt+backspace	恢复上一次删除

3．标点

在 MATLAB 语言中，一些标点符号也被赋予了特殊的意义，或代表一定的运算，具体内容如表 1-7 所示。

表 1-7　　MATLAB 语言的标点

标点	说明	标点	说明
：	冒号，具有多种应用功能	%	百分号，注释标记
；	分号，区分行及取消运行结果显示	！	惊叹号，调用操作系统运算
，	逗号，区分列及函数参数分隔符	=	等号，赋值标记
()	括号，指定运算的优先级	‘	单引号，字符串的标示符
[]	方括号，定义矩阵	.	小数点及对象域访问
{}	大括号，构造单元数组	…	续行符号

4．搜索路径与扩展

当 MATLAB 7.0 对函数或文件等进行搜索时，都是在其搜索路径下进行的。如果用户调用的函数在搜索路径之外，MATLAB 7.0 则认为此函数并不存在。一般情况下，MATLAB 7.0 系统的函数（包括工具箱函数）都在系统默认的搜索路径之中，但是用户自己书写的函数有可能并没有保存在搜索路径下。要解决这个问题，只需把程序所在的目录扩展成 MATLAB 7.0 的搜索路径即可。下面就向读者介绍设置 MATLAB 7.0 搜索路径及其扩展的方法。

（1）查看 MATLAB 7.0 的搜索路径

可以通过菜单命令和 path、genpath 命令函数两种方法来查看搜索路径。

选择 MATLAB 7.0 主窗口中的“File→Set Path”菜单，进入“设置搜索路径”对话框，如图 1-31 所示。通过该对话框可为 MATLAB 添加或删除搜索路径，用户只要稍加练习就可掌握。

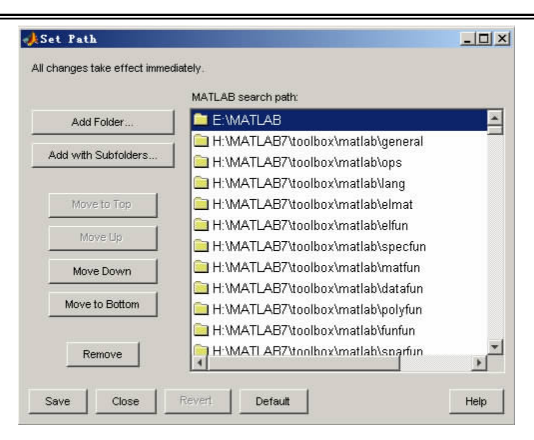


图 1-31 "设置搜索路径"对话框

在命令窗口中输入命令 path 或 genpath 可得到 MATLAB 7.0 的所有搜索路径，代码如下：

```
>> path
MATLABPATH

E:\MATLAB
H:\MATLAB7\toolbox\MATLAB\general
H:\MATLAB7\toolbox\MATLAB\ops
H:\MATLAB7\toolbox\MATLAB\lang
…
H:\MATLAB7\work

>>genpath

ans =

H:\MATLAB7\toolbox;H:\MATLAB7\toolbox\aeroblks;H:\MATLAB7\toolbox\aeroblks\aeroblks;H:\MATLAB7\toolbox\aeroblks\aeroblks\ja;H:\MATLAB7\toolbox\aeroblks\aerodemos;H:\MATLAB7\toolbox\aeroblks\aerodemos\texture;H:\MATLAB7\toolbox\bioinfo;H:\MATLAB7\toolbox\bioinfo\biodemos;H:\MATLAB7\toolbox\bioinfo\biodemos\html;H:\MATLAB7\toolbox\bioinfo\biodemos\ja;H:\MATLAB7\toolbox\bioinfo\bioinfo;H:\MATLAB7\toolbox\bioinfo\bioinfo\ja;
…//结果生成一个字符串
```

（2）设置 MATLAB 7.0 的搜索路径

方法一：在 MATLAB 7.0 命令窗口中输入 editpath 或 pathtool 命令或通过"File→Set Path"菜单，进入如图 1-31 所示的"设置搜索路径"对话框，然后通过该对话框可以编辑搜索路径。

方法二：在命令窗口执行"path(path, 'e:\MATLAB')"，然后通过图 1-31 所示的"设置搜索路径"对话框查看"e:\MATLAB"是否在搜索路径中。

方法三：在命令窗口执行"addpath　e:\MATLAB　-end"，将新的目录加到整个搜索路径的末尾。如果将 end 改为 begin，可以将新的目录加到整个搜索路径的开始。

1.6 使用 MATLAB 7.0 帮助系统

MATLAB 7.0 为用户提供了非常完善的帮助系统，例如 MATLAB 7.0 的在线帮助、帮助窗口、帮助提示、HTML 格式的帮助、pdf 格式的帮助文件以及 MATLAB 7.0 的示例和演示等。通过使用 MATLAB 7.0 的帮助菜单或在命令窗口中输入帮助命令，可以很容易地获得 MATLAB 7.0 的帮助信息，并能通过帮助进一步学习 MATLAB 7.0。下面分别介绍 MATLAB 7.0 中三种类型的帮助系统。

1．命令窗口查询帮助系统

常见的帮助命令如表 1-8 所示。

表 1-8　常用 MATLAB 7.0 帮助命令

帮助命令	功能	帮助命令	功能
help	获取在线帮助	which	显示指定函数或文件的路径
demo	运行 MATLAB 7.0 演示程序	lookfor	按照指定的关键字查找所有相关的 M 文件
tour	运行 MATLAB 7.0 漫游程序	exist	检查指定变量或文件的存在性
who	列出当前工作空间中的变量	helpwin	运行帮助窗口
whos	列出当前工作空间中变量的更多信息	helpdesk	运行 HTML 格式帮助面板 Help Desk
what	列出当前目录或指定目录下的 M-文件、MAT 文件和 MEX 文件	doc	在网络浏览器中显示指定内容的 HTML 格式帮助文件，或启动 helpdesk

【Help 命令】：在命令窗口用于显示 MATLAB 7.0 函数的帮助。调用格式如下：

```
Help                      //在命令窗口列出所有主要的基本帮助主题
help /                    //列出所有运算符和特殊字符
help functionname         //在命令窗口列出 functionname M-文件的描述及语法
help toolboxname          //在命令窗口列出 toolboxname 文件夹中的内容
help toolboxname/functionname
help classname.methodname    //显示某一类的函数帮助
help classname
help syntax
t = help('topic')
```

【示例 1】

```
>> help
HELP topics

MATLAB\general          -  General purpose commands.
MATLAB\ops              -  Operators and special characters.
MATLAB\lang             -  Programming language constructs.
…
kernel\embedded         -  xPC Target Embedded Option
```

```
MATLAB7\work              -   (No table of contents file)
e:\MATLAB                 -   (No table of contents file)
```

【示例 2】

```
>> help add
 --- help for hgbin/add.m ---
 HGBIN/ADD Add method for hgbin object
     This file is an internal helper function for plot annotation.
     There is more than one add available.    See also
         help ccshelp/add.m
         help iviconfigurationstore/add.m
         help cgrules/add.m
         help des_constraints/add.m
         help xregcardlayout/add.m
         help xregcontainer/add.m
         help xregmulti/add.m
         help cgddnode/add.m

     Reference page in Help browser
         doc add
```

【lookfor 命令】：按照指定的关键字查找所有相关的 M 文件。其调用格式如下：

```
lookfor topic
lookfor topic –all
```

【示例】

```
>> lookfor    inverse
INVHILB Inverse Hilbert matrix.
IPERMUTE Inverse permute array dimensions.
ACOS      Inverse cosine.
ACOSD    Inverse cosine, result in degrees.
…
ADDINVG Add the inverse Gaussian distribution.
STDRINV Compute inverse c.d.f. for Studentized Range statistic
```

2．联机帮助系统

MATLAB 7.0 的联机帮助系统非常全面。用户可以通过下面介绍的方法进入 MATLAB 7.0 的联机帮助系统（如图 1-32 所示）。

- 直接单击 MATLAB 7.0 主窗口中的“?”按钮；
- 选中 Help 菜单的前 4 项中的任意一项；
- 在命令窗口中执行 helpwin、helpdesk 或 doc。

下面介绍联机帮助系统的使用方法和技巧。

联机帮助系统界面的菜单项与大多数 Windows 程序界面的菜单含义和用法都差不多，熟

悉 Windows 的用户可以很容易地掌握，在此不作详细介绍。帮助向导页面包含 4 个页面，分别是帮助主题（Contents）、帮助索引（Index）、查询帮助（Search）以及演示帮助（Demos）。如果知道需要查询的内容的关键字，一般可选择 Index 或 Search 模式来查询；只知道需要查询的内容所属的主题或是只是想进一步了解和学习某一主题，一般可选择 Contents 或 Demos 模式来查询。

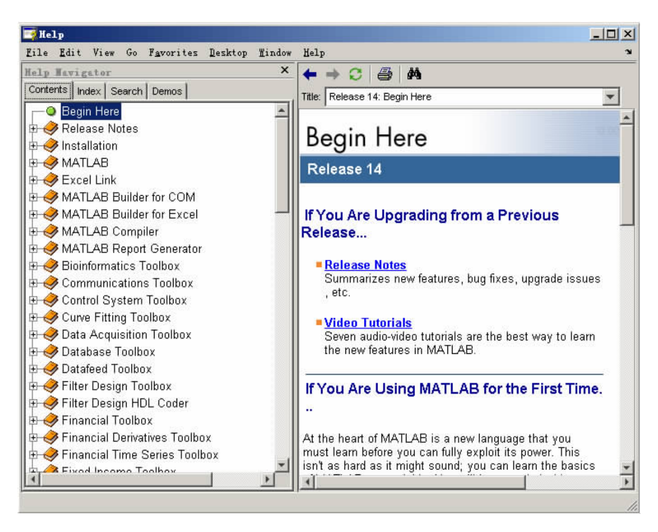


图 1-32　联机帮助系统

3．联机演示系统

通过联机演示系统，用户可以直观、快速地学习 MATLAB 7.0 中某个工具箱的使用方法，它是有关的参考书籍不能替代的。下面就向读者介绍如何使用演示系统。

可以通过以下方式打开联机演示系统。

- 选择 MATLAB 7.0 主窗口菜单的“Help→Demos”选项；
- 在命令窗口输入 demos；
- 直接在帮助页面上选择 Demos 页。

【示例】

- 在 MATLAB 7.0 命令窗口中执行“>>demos”命令，弹出如图 1-33 所示的界面。

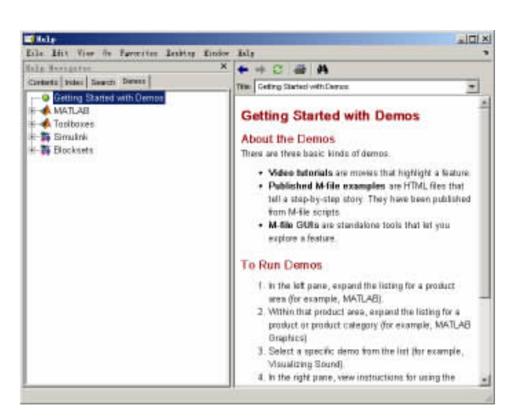


图 1-33　联机演示系统

- 在 Demos 页面中选择“Signal Processing”工具箱中的“Spectral Analysis and Statistical Signal Processing”选项（如图 1-34 所示）。

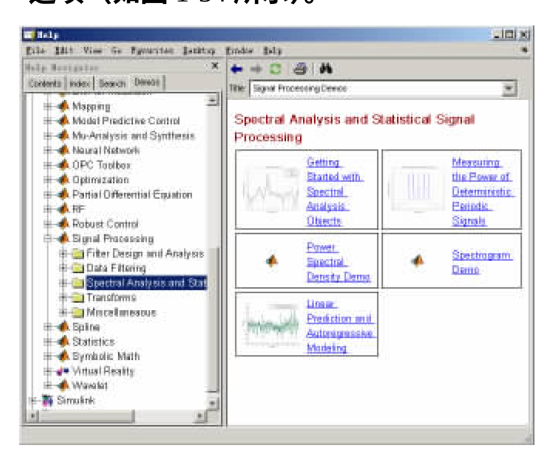


图 1-34　演示示例

- 然后在右边窗口中选择“Power Spectral Density Demo”选项打开对话框。用鼠标单击页面底部有下划线的文字“Run this demo”，打开如图 1-35 所示的示例界面窗口。在该窗口中，用户可以选择不同的信号，窗口中就会自动显示该信号的功率谱密度。

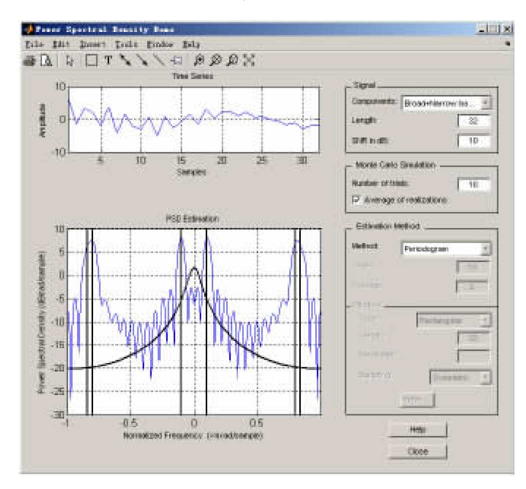


图 1-35　“功率谱密度的演示示例界面”窗口

联机演示系统对于学习工具箱以及 MATLAB 7.0 各个方面的应用的用户非常有意义。通过演示示例，用户可以快速直观地掌握某一工具的使用方法，而不必从枯燥的理论开始学起。

1.7　初识 MATLAB

下面以一个简单的示例向读者展示如何使用 MATLAB 7.0 进行简单的数值计算。

- 用鼠标左键双击桌面上的 MATLAB 7.0 图标，进入 MATLAB 7.0 的工作环境界面（如图 1-14 所示）；
- 在命令窗口中输入“>>w=2*pi;”，按 Enter 键，可以在工作空间窗口看到变量 *w*，大

小为 6.2832**；**

- **在命令窗口中输入"** >>y= sin(w*2/3); **"，按** Enter **键，可以在工作空间窗口看到变量** *y*，**大小为**-0.86603**；**
- **在命令窗口中输入"** >>z=sin(w/3) **"，按** Enter **键，可以在工作空间窗口看到变量** *z*，**大小为** 0.86603，**命令窗口中显示代码如下：**

```
z =
    0.8660
```

- **在命令窗口中输入如下代码：**

```
>> w2=pi
m=sin(w2/2)
```

- **按** Enter **键，可以在工作空间窗口看到变量** *w*2 **和** *m*，**大小分别为** 3.1416 **和** 1，**命令窗口中显示代码如下：**

```
w2 =
    3.1416
m =
     1
```

注意：当命令后面有分号时，按 Enter **键后，命令窗口中不显示运算结果；如果无分号，则在命令窗口中显示运算结果。当希望先输入多条语句，然后再同时执行它们，则在输入下一条命令时，要按住** Ctrl **键的同时按下** Enter **键进行换行输入。**

第 2 章　MATLAB 基础知识

本章介绍了 MATLAB 7.0 的一些基础知识，包括 MATLAB 的数据类型、基本矩阵操作、运算符和字符串处理函数。本章的知识是 MATLAB 7.0 编程的基础，熟练掌握了本章知识有利于学习后面的章节。

2.1　数据类型

MATLAB 7.0 中定义了很多种数据类型，包括整数、浮点数、字符、字符串和逻辑类型等。用户甚至可以定义自己的数据类型。本小节讨论各种数据类型以及在 MATLAB 7.0 中使用它们的方法。

在 MATLAB 7.0 中有 15 种基本数据类型。每种基本的数据类型均以矩阵的形式出现，该矩阵可以是最小的 0×0 矩阵到任意大小的 *n* 维矩阵。数据类型结构如图 2-1 所示，其中带下划线的为基本数据类型。下面将依次介绍各种数据类型的用法。

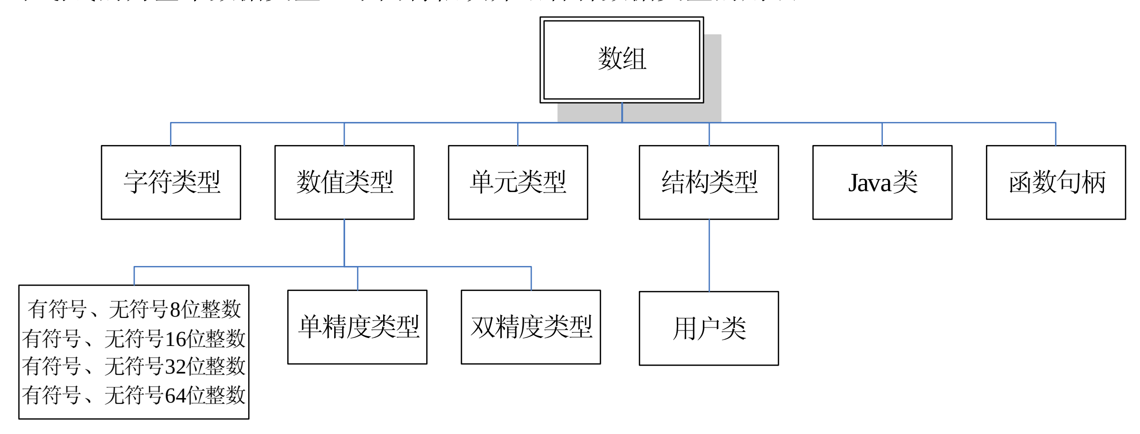


图 2-1　数据类型结构图

2.1.1　数值类型

数值类型包含整数、浮点数和复数 3 种类型。另外 MATLAB 7.0 还定义了 inf 和 NaN 两个特殊数值。

1．整数类型

在 MATLAB 7.0 中整数类型包含 4 种有符号整数和 4 种无符号整数。有符号整数可以用来表示负数、零和正整数，而无符号整数则只可以用来表示零和正整数。MATLAB 7.0 支持 1、2、4 和 8 字节的有符号整数和无符号整数。这 8 种数据类型的名称、表示方法和类型转

换函数如表 2-1 所示。应用时要尽可能用字节数少的数据类型表示数据，这样可以节约存储空间和提高运算速度。例如最大值为 100 的数据可以用 1 个字节的整数来表示，而没有必要用 8 个字节的整数来表示。

表 2-1　整数的数据类型和表示范围

数据类型名称	数据类型表示范围	类型转换函数
有符号 1 字节整数	-2^{7} - 2^{7}-1	int8()
有符号 2 字节整数	-2^{15} - 2^{15}-1	int16()
有符号 4 字节整数	-2^{31} - 2^{31}-1	int32()
有符号 8 字节整数	-2^{63} - 2^{63}-1	int64()
无符号 1 字节整数	0 - 2^{8}-1	uint8()
无符号 2 字节整数	0 - 2^{16}-1	uint16()
无符号 4 字节整数	0 - 2^{32}-1	uint32()
无符号 8 字节整数	0 - 2^{64}-1	uint64()

在表 1-2 中的类型转换函数可以用于把其他数据类型的数值强制转换为整数类型。此外类型转换函数还可以用于生成整数类型的数值。例如，如果需要产生一个有符号 2 字节整数的数值可以用如下语句实现，具体代码如下：

```
x = int16(12531);
```

2．浮点数类型

MATLAB 7.0 有双精度浮点数和单精度浮点数两种浮点数。双精度浮点数为 MATLAB 7.0 默认的数据类型。如果某个数据没有被显式地指定数据类型，那么 MATLAB 7.0 会用双精度浮点数来存储它。为了得到其他类型的数值类型，可以使用类型转换函数。

MATLAB 7.0 中的双精度浮点数和单精度浮点数均采用 IEEE 754 中规定的格式来定义。其表示范围、存储大小和类型转换函数如表 2-2 所示。

表 2-2　浮点数的数据类型和表示范围

数据类型名称	存储大小	表示范围	类型转换函数
双精度浮点数	4 字节	$-1.79769*10^{308}$~$+1.79769*10^{308}$	double()
单精度浮点数	8 字节	$-3.40282*10^{38}$~$+3.40282*10^{38}$	single()

3．复数类型

复数包含独立的两部分，即实部和虚部。虚部的单位是-1 的开平方根，在 MATLAB 7.0 中可以用 i 或者 j 来表示。

可以用如下赋值语句来产生复数：

```
a=5+10i;
```

也可以用函数 complex 来产生复数，示例代码如下：

```
x=5;
y=10;
z=complex(x,y);
```

其中 x、y 为实数，得到的 z 是以 x 为实部，y 为虚部的复数。

也可以这样使用 complex 函数，具体代码如下：

```
x=5;
```

```
z=complex(x);
```

其中 *x* 为实数，得到的 *z* 是以 *x* 为实部，以 0 为虚部的复数。

4．inf 和 NaN

MATLAB 7.0 中规定用 inf、-inf 来表示正无穷大和负无穷大。除法运算中除数为 0 或者运算结果溢出都会导致 inf 或-inf 的结果。以下 3 条语句运算产生的结果均为 inf，具体代码如下：

```
2/0
x = exp(3000)
x = log(0)
```

MATLAB 7.0 中规定用 NaN 来表示一个既不是实数也不是复数的数值。NaN 是 Not a Number 的缩写。类似 0/0、inf/inf 这样的表达式得到的结果均为 NaN。

2.1.2 逻辑类型

逻辑类型用 1 和 0 来表示 true 和 false 两种状态。可以用函数 logical()来得到逻辑类型的数值。函数 logical()可以把任何非零的数值转换为逻辑 true（即 1），把数值 0 转换为逻辑 false（即 0）。示例代码如下：

```
logical(-1)
```

上述语句得到代码如下：

```
Warning: Values other than 0 or 1 converted to logical 1
ans =         1
```

在 2.3.2 小节中将介绍的逻辑关系运算符也可以得到逻辑类型的数据。

2.1.3 字符和字符串

MATLAB 7.0 中规定用数据类型 char 来表示一个字符。一个 char 类型的 1×n 数组则可以称为字符串 string。MATLAB 7.0 中 char 类型都是以 2 字节的 unicode 字符来存储的。

可以用一对单引号来表示字符串，例如下面的代码：

```
str='I am a great person';
```

也可以用 char 函数来构造一个字符串，例如下面的代码：

```
str=char('[65 66]');
```

上述语句得到字符串'AB'。

关于字符串的更深入的介绍请参考本书 2.4 小节。

2.1.4 函数句柄

函数句柄是 MATLAB 7.0 中用来提供间接调用函数的数据类型。函数句柄可以转递给其他函数以便该函数句柄所代表的函数可以被调用。函数句柄还可以被存储起来，以便以后利用。

函数句柄可以用符号@后面跟着函数名来表示，例如下面的代码：

```
fhandle=@sin;
```

sin 为 MATLAB 7.0 中自带的正弦函数，得到的输出变量 *fhandle* 为 sin 函数的句柄。可以利用 *fhandle* 来调用 sin 函数，例如下面的代码：

```
fhandle(0)
```

上面语句得到的输出代码如下：

```
ans =        0
```

实际上，该程序中的语句 fhandle(0)相当于语句 sin(0)。

2.1.5　结构体类型

结构体是根据属性名组织起来的不同类型数据的集合。有一种容易与结构体类型混淆的数据类型是单元数组类型，它是一种特殊类型的 MATLAB 7.0 数组，它的每一个元素叫做单元，而每一个单元包含 MATLAB 7.0 数组。结构体和单元数组的共同之处在于它们都提供了一种分级存储机制来存储不同类型的数据，不同之处是组织数据的方式不一样。结构体数组里的数据是通过属性名来引用的，而在单元数组里，数据是通过单元数组下标引用来操作的。本节将介绍结构体数组，而单元数组将在 2.1.6 小节中介绍。

结构体数组是一种由“数据容器”组成的 MATLAB 7.0 数组，这种“数据容器”称为结构体的属性（field）。结构体的任何一个属性可以包含任何一种类型的数据。如图 2-2 所示是一个结构体，它有所 3 个属性，即 Name、Score 和 Salary，其中 Name 是一个字符串，Score 是一个标量，Salary 是一个 1×5 的向量。

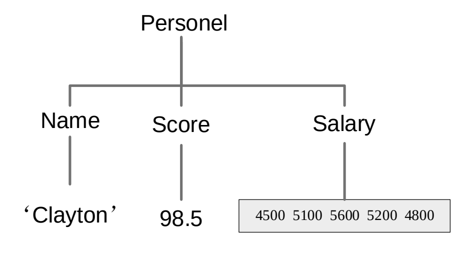


图 2-2　结构体的属性

和其他的数据类型一样，结构体也是一种数组。一个单独的结构体就是一个 1×1 的结构体数组。用户可以构造任意维数和形状的结构体数组，当然也包含多维结构体数组。

1．结构体数组的构造

构造一个结构体数组有如下两种方法：

- 利用赋值语句；
- 利用函数 struct()来进行定义。

下面就通过为结构体中的每一个属性赋值来构造一个结构体数组。例如，建立如图 2-2 所示的结构体数组 *Personel*，可以用如下语句：

```
Personel.Name='Clayton';
Personel.Score=98.5;
```

```
Personel.Salary=[4500 5100 5600 5200 4800];
Personel                                   %在命令行输出 Personel 变量的信息
```

上述语句得到输出代码如下：

```
Personel =
      Name: 'Clayton'
     Score: 98.5000
    Salary: [4500 5100 5600 5200 4800]
```

还可以用如下语句把结构体数组扩展成 1×2 的结构体，代码设置如下：

```
Personel(2).Name='Dana';
Personel(2).Score=100;
Personel(2).Salary=[6700 9000];
```

由上述语句使结构体数组 *Personel* 的维数变为 1×2。当用户扩展结构体数组时，MATLAB 7.0 对未指定数据的属性自动赋值成空矩阵，使其满足以下规则：

- 数组中的每个结构体都具有同样多的属性名；
- 数组中的每个结构体都具有相同的属性名。

例如，下面语句使结构体数组 *Personel* 的维数变为 1×3，此时 *Personel(3).Score* 和 *Personel(3). Salary* 由于未指定数据，MATLAB7.0 将其设为空矩阵。

```
Personel(3).Name='John';
```

要注意的是结构体数组中元素属性的大小并不要求一致，例如结构数组 *Personel* 中的 *Name* 属性和 *Salary* 属性都具有不同的长度。

除了使用赋值语句来构造结构数组外，还可以用函数 struct()来实现构造结构数组。函数 struct 的基本调用格式为：

```
strArray = struct('field1',val1,'field2',val2, ...)
```

上面语句中的输入变量为属性名和相应的属性值。

函数 struct()可以有不同的调用方法来实现构造结构体矩阵，例如要实现一个 1×3 的结构数组 *Personel* 的方法如表 2-3 所示。

表 2-3　　使用 struct 函数的方法

方法	调用格式	初始值状况
单独使用 struct 函数	*Personel(3)*=struct('Name','John','Score',85.5,'Salary',[4500 4200])	*Personel(1)* 和 *Personel(2)* 的属性值都是空矩阵，*Personel(3)* 的值如输入
struct 函数与 repmat 函数配合使用	repmat(struct('Name','John','Score',85.5,'Salary',[4500 4200]),1,3)	数组的所有元素具有和输入一样的值
struct 函数的输入为单元数组	struct('Name',{'Clayton','Dana','John'},'Score',{98.5,100,85.5},'Salary',{[4500 4200],[],[]})	结构数组的属性值由单元数组指定

2．访问结构体数组的数据

使用结构体数组的下标引用，可以访问结构体数组任何元素及其属性。同样也可以给任何元素及其属性赋值。例如有一个结构体数组如图 2-3 所示，它可以通过下面语句来生成：

```
Personel=struct('Name',{'Clayton','Dana','John'},'Score',{98.5,100,[]},'Salary',{[4500 5100 5600 5200 4800],[6700 9000],[]})
```

用户可以访问结构数组的任意子数组。例如，下面的命令行生成一个 1×2 的结构数组：

```
NewPersonel= Personel(1:2)
```

上述语句得到输出代码如下：

```
NewPersonel =
1x2 struct array with fields:
    Name
    Score
    Salary
```

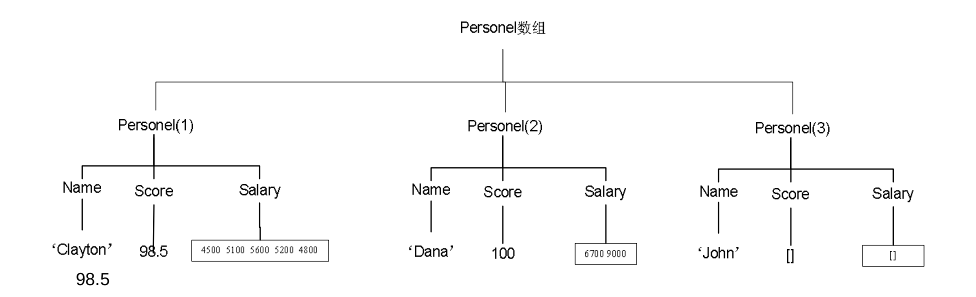


图 2-3　结构数组

如果要访问结构体数组的某个元素的某个属性，可以用如下格式：

```
Personel(2).Name
```

上述语句得到输出代码如下：

```
ans =
Dana
```

如果要访问结构体数组的某个元素的某个属性的元素值，可以使用如下格式：

```
Personel(1).Salary(3)
```

上述语句得到的输出代码如下：

```
ans =
        5600
```

如果想得到结构体数组的所有元素的某个属性值，可以使用如下格式：

```
Personel.Name
```

上述语句得到输出代码如下：

```
ans =
Clayton
ans =
Dana
```

```
ans =
John
```

以上结果表明 Personel.Name 格式的输入将返回结构体数组的所有元素的属性值。可以使用矩阵合并符[]来合并这些结果，程序语句如下：

```
Salary=[Personel.Salary]
```

上述语句得到输出代码如下：

```
Salary =
  4500    5100    5600    5200    4800    6700    9000
```

也可以用把它们合并在一个单元数组里，代码设置如下：

```
Salary={Personel.Salary}
```

上述语句得到输出代码如下：

```
Salary =
    [1x5 double]    [1x2 double]    []
```

2.1.6　单元数组类型

单元数组就是每个元素为一个单元的数组。每个单元都可以包含任意数据类型的 MATLAB 7.0 数组。例如，单元数组的一个单元可以是一个实数矩阵，或是一个字符串数组，也可以是一个复向量数组。图 2-4 所示是一个 2×3 的单元数组，可见其每一个单元内的数据类型均不相同。

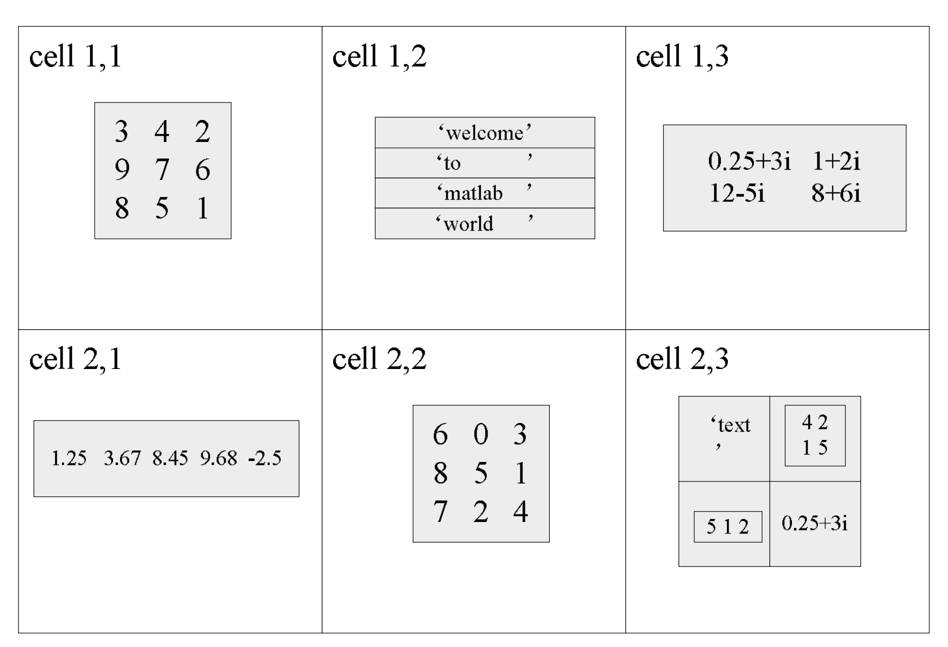


图 2-4　单元数组示意图

1．单元数组的构造

构造单元数组有左标志法和右标志法两种方法。下面就详细介绍这两种方法。

- *左标志法*

左标志法就是把单元标志{}放在左边，例如，创建一个 2×2 的单元数组可以使用如下语句：

```
c{1,1}='Clayton';
c{1,2}=eye(3,3);
c{2,1}=@sin;
c{2,2}=true;
```

- *右标志法*

右标志法就是把标志符放在右边，例如，创建和上面一样的单元数组可以使用如下语句：

```
c(1,1)={'Clayton'};
c(1,2)={eye(3,3)};
c(2,1)={@sin};
c(2,2)={true};
```

上述语句还可以简单地写为下面的代码：

```
c={'clayton',eye(3,3);@sin,true}
```

要显示单元数组可以在命令行中直接输入单元数组的名字，代码设置如下：

```
c
```

由上述语句得到输出代码如下：

```
c =
    'Clayton'     [3x3 double]
    [sin    ]     [          1]
```

另一种显示单元数组的方法是使用函数 celldisp()，代码如下：

```
celldisp(c)
```

由上述语句得到的输出代码如下：

```
c{1,1} =
clayton
c{2,1} =
    @sin
c{1,2} =
     1     0     0
     0     1     0
     0     0     1
c{2,2} =
     1
```

值得注意的是函数 celldisp()的显示格式与直接输入单元数组名的显示格式是不同的。celldisp()函数更适用于具有大量数据的单元数组的显示。

2．单元数组的读取

以上小节中的单元数组 *c* 作为例子。要读取 *c{1,1}*中的字符串，可以使用如下语句：

```
Str= c{1,1}
```

由上述语句得到输出代码如下：

```
Str =
clayton
```

如果要读取单元数组的若干个单元的数据，例如读取单元数组 *c* 的第二行，可以用下面语句：

```
c(2,:)
```

上述语句得到输出代码如下：

```
ans =
      [sin]      [1]
```

3．**单元数组的删除**

将空矩阵赋给单元数组的某一整行或者某一整列，就可以删除单元数组的这一行或者一列。

例如，删除单元数组 *c* 的第一行可以用如下语句：

```
c(1,:)=[]
```

上述语句得到输出代码如下：

```
c =
      [sin]      [1]
```

2.2 基本矩阵操作

MATLAB 7.0 是基于矩阵运算的一个软件，所有的数据均以二维矩阵或高维矩阵的形式存储，因此 MATLAB 7.0 又被称为矩阵实验室。

MATLAB 7.0 中最基本的数据结构是二维的矩阵。二维的矩阵可以方便地存储和访问大量数据。每个矩阵的单元可以是数值类型、逻辑类型、字符类型或者其他任何的 MATLAB 7.0 数据类型。无论是单个数据还是一组数据，MATLAB 7.0 均采用二维的矩阵来存储。对于一个数据，MATLAB 7.0 用 1×1 矩阵来表示；对于一组数据，MATLAB 7.0 用 1×*n* 矩阵来表示，其中 *n* 是这组数据的长度。MATLAB 7.0 也支持多维的矩阵，MATLAB 7.0 中称这类数据为多维数组（array）。

为了方便，把 1×1 的矩阵称为标量，把 1×n 的矩阵称为向量。把至少有一维的长度为 0 的矩阵称为空矩阵，空矩阵可以用[]来表示。

例如，实数 1.5 是 1×1 的双精度浮点数类型矩阵。在 MATLAB 7.0 中可以用语句 whos 来显示数值的数据类型和储存矩阵大小。以下语句可以用于查看实数 1.5 的数据类型和储存矩阵大小：

```
a=1.5;
whos a      %显示变量 a 的信息
```

由上述语句得到输出代码如下：

```
Name        Size                    Bytes  Class
a           1x1                         8  double array
```

例如，字符串 'I am a great person'是 1×19 的字符矩阵类型矩阵，可以用如下语句来查看该字符串的数据类型和储存矩阵大小：

```
str='I am a great person';
whos str
```

由上述语句得到输出代码如下：

```
Name        Size                    Bytes   Class
str         1x19                     38     char array
```

2.2.1　矩阵的构造

1．简单矩阵构造

最简单的构造矩阵方法是采用矩阵构造符[]，构造一行的矩阵可以把矩阵元素放在矩阵构造符[]中，并以空格或者逗号来隔开它们，其代码设置如下：

```
row = [E1, E2, ..., Em]     或者     row = [E1 E2 ... Em]
```

例如，一个 1×4 的矩阵可以通过如下语句得到：

```
a=[1 2 3 4]
```

或者是下面的代码：

```
a=[1,2,3,4]
```

如果矩阵是多行的，行与行之间用分号隔开，其代码设置如下：

```
A = [row1; row2; ...; rown]
```

例如一个 3×4 的矩阵可以用如下语句得到：

```
A=[1,2,3,4;5,6,7,8;9,10,11,12]
```

上述语句得到矩阵 ***A*** 如下：

```
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

2．特殊矩阵构造

MATLAB 7.0 还提供一些函数用来构造一些特殊的矩阵，这些函数如表 2-4 所示。

表 2-4　　特殊矩阵函数

函数名	函数用途	基本调用格式
ones	产生矩阵元素全为 1 的矩阵	*A*=ones(*n*)　产生 *n*×*n* 的 1 *A*=ones(*m*,*n*)　产生 *m*×*n* 的 1
zeros	产生矩阵元素全为 0 的矩阵	*A*=zeros(*n*)　产生 *n*×*n* 的 0 *A*=zeros(*m*,*n*)　产生 *m*×*n* 的 0
eye	产生单位矩阵，即主对角线上的元素为 1，其他元素全为 0	*A*=eye(*n*)　产生 *n*×*n* 的单位矩阵 *A*=eye (*m*,*n*)　产生 *m*×*n* 的单位矩阵

续表

函数名	函数用途	基本调用格式	
Diag	把向量转化为对角矩阵或者得到矩阵的对角元素	*X*=diag(*v*,*k*) *X*=diag(*v*) *v*=diag(*X*,*k*) *v*=diag(*X*)	把向量 *v* 转换为一个对角矩阵。 把向量 *v* 转换为一个主对角矩阵。 得到矩阵 ***X*** 的对角元素 得到矩阵 ***X*** 的主对角元素
magic	产生魔方矩阵，即每行、每列之和相等的矩阵	magic(*n*)	产生 *n*×*n* 的魔方矩阵
rand	产生 0~1 均匀分布的随机数	*Y* = rand(*n*) *Y* = rand(*m*,*n*)	产生 *n*×*n* 的 0-1 均匀分布的随机数 产生 *m*×*n* 的 0-1 均匀分布的随机数
randn	产生均值为 0，方差为 1 高斯分布的随机数	*Y* = randn(*n*) *Y* = randn(*m*,*n*)	产生 *n*×*n* 的标准高斯分布的随机数 产生 *m*×*n* 的标准高斯分布的随机数
randperm	产生整数 1~n 的随机排列	*p* = randperm(*n*)	产生整数 1 到 *n* 的随机排列
compan	产生多项式的伴随矩阵	*A* = compan(*u*)	产生多项式 *u* 的伴随矩阵

例如要产生一个 3×4 的全 0 矩阵，可以采用函数 zeros()实现，代码设置如下：

```
a=zeros(3,4)
```

由上述语句得到矩阵：

```
a =
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

例如要产生一个 1×5 的 0~1 均匀分布的随机数，实现方法如下：

```
a=rand(1,10)
```

由上述语句得到矩阵：

```
a =
    0.4186    0.8462    0.5252    0.2026    0.6721
```

值得注意的是函数 rand()是以机器时间作为随机种子的，每次运行上述语句得到的结果都是不同的。

3．向量的构造

向量在构造矩阵和对矩阵进行索引时是很有用的，因此 MATLAB 7.0 提供了专门用于产生向量的运算符——“:”。

冒号可以用来构造步长为 1 的递增向量，其格式为：

```
a:b
```

上式产生从 ***a~b*** 的以步长为 1 递增向量。例如：

```
A = 1:5
```

上述语句生成的结果是：

```
A =     1     2     3     4     5
```

向量中的元素不必是正整数，可以是任何整数。例如下面的代码：

```
A=-3.2:2.2
```

由上述语句生成的结果是：

```
A =    -3.2000    -2.2000    -1.2000    -0.2000     0.8000     1.8000
```

在默认的情况下，MATLAB 7.0 总是保证步长精确地为 1，即使在 *b* 值不是距离 *a* 值整数距离时也如此。例如下面的代码：

```
A=1:5.2
```

由上述语句生成的结果是：

```
A =    1    2    3    4    5
```

如果 *b* 值小于 *a* 值，则 MATLAB 7.0 返回一个空矩阵。例如下面的代码：

```
A=3:1
```

由上述语句生成的结果是：

```
A =    Empty matrix: 1-by-0
```

MATLAB 7.0 也支持产生任意步长的向量，步长甚至可以是负数。例如下面的代码：

```
y=0:pi/4:pi
```

上面语句产生从 0 开始到 pi 结束的以 1/4*pi 为步长的向量。由上述语句生成的结果是：

```
y =         0    0.7854    1.5708    2.3562    3.1416
```

例如：

```
z=5:-1:1
```

上面语句产生从 5 开始到 0 结束的以-1 为步长的向量。上述语句生成的结果是：

```
z =    5    4    3    2    1
```

能够构造向量的其他函数还有 linspace()函数和 logspace()函数。Linspace()函数用于创建指定长度的等距向量。例如下面的代码：

```
A=linspace(-6,6,4)
```

上述语句产生-6，6 之间等间距的 4 个点。由上述代码生成的结果是：

```
A =    -6    -2    2    6
```

logspace()函数用于创建对数等距的向量，其使用方法与 linspace()函数完全相同。

2.2.2　矩阵大小的改变

1．矩阵的合并

矩阵的合并就是把两个或者两个以上的矩阵数据连接起来得到一个新的矩阵。前面介绍的矩阵构造符[]不仅可用于构造矩阵，同时还可以作为一个矩阵合并操作符。表达式 *C*=[*A B*]在水平方向合并矩阵 ***A*** 和 ***B***，而表达式 *C*=[*A*;*B*]在竖直方向合并矩阵 ***A*** 和 ***B***。

例如：

```
a=ones(2,3);
b=zeros(2,3);
c=[a;b]
```

由上述语句得到结果代码如下：

```
c =
     1     1     1
     1     1     1
     0     0     0
```

```
     0     0     0
```

例如下面的代码：

```
a=ones(2,3);
b=zeros(2,3);
c=[a b]
```

由上述语句得到结果代码如下：

```
c =
     1     1     1     0     0     0
     1     1     1     0     0     0
```

可以用矩阵合并符来构造任意大小的矩阵。不过需要注意的是在矩阵合并的过程中一定要保持矩阵的形状是方形，否则矩阵合并将无法进行。图 2-5 表明具有相同高度的两个矩阵可以在水平方向合并为一个新的矩阵。而图 2-6 则表明不具有相同高度的两个矩阵，不允许合并为一个矩阵。

1	2
3	4
5	6

3×2

＋

4	5	6
7	8	9
10	11	12

3×3

＝

1	2	4	5	6
3	4	7	8	9
5	6	10	11	12

3×5

图 2-5　正确的矩阵合并

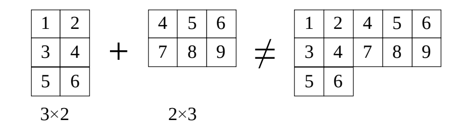


图 2-6　不正确的矩阵合并

除了使用矩阵合并符[]来合并矩阵外，还可以使用矩阵合并函数来合并函数。这些矩阵合并函数的函数描述和基本调用格式如表 2-5 所示。

表 2-5　　矩阵合并函数

函数名	函数描述	基本调用格式
cat	在指定的方向合并矩阵	cat(*DIM*,*A*,*B*)　在 *DIM* 维方向合并矩阵 ***A*,*B*** cat(*2*,*A*,*B*)　与[***A B***]用途一致 cat(*1*,*A*,*B*)　与[***A*;*B***] 用途一致
horzcat	在水平方向合并矩阵	horzcat(*A*,*B*)　与[***A B***]用途一致
vertcat	在竖直方向合并矩阵	vertcat(*A*,*B*)　与[***A*;*B***]用途一致
repmat	通过复制矩阵来构造新的矩阵	*B* = repmat(*A*,*M*,*N*)　得到 ***M´N*** 个 ***A*** 的大矩阵
blkdiag	用已知矩阵来构造块对角化矩阵	*Y* = blkdiag(*A*,*B*,...)　得到以矩阵 ***A***、***B***…等为对角块的矩阵 ***Y***

例如下面的代码：

```
a=[1 2;3 4];
b=repmat(a,2,3)
```

由上述语句得到输出代码如下：

```
b =
     1     2     1     2     1     2
```

```
        3     4     3     4     3     4
        1     2     1     2     1     2
        3     4     3     4     3     4
```

又例如下面的代码：

```
a=eye(2)*3;
b=magic(3);
c=blkdiag(a,b)
```

由上述语句得到输出代码如下：

```
c =
        3     0     0     0     0
        0     3     0     0     0
        0     0     8     1     6
        0     0     3     5     7
        0     0     4     9     2
```

2．矩阵行列的删除

要删除矩阵的某一行或者是某一列，只要把该行或者该列赋予一个空矩阵[]即可。例如有一个 4×4 的魔方矩阵，代码设置如下：

```
A = magic(4)
```

上述语句得到矩阵 ***A***：

```
A =
       16     2     3    13
        5    11    10     8
        9     7     6    12
        4    14    15     1
```

如果想删除矩阵的第 2 行，则可以用如下语句：

```
A(2,:)=[]
```

由上述语句得到新的矩阵 ***A*** 如下：

```
A =
       16     2     3    13
        9     7     6    12
        4    14    15     1
```

2.2.3　矩阵下标引用

本小节将介绍用矩阵下标来读取和改写矩阵元素的值的方法。包括如何用矩阵下标访问单个矩阵阵元素、如何线性引用矩阵元素、引用矩阵元素方式转换和访问多个矩阵元素。

1．矩阵下标访问单个矩阵阵元素

若 *A* 是一个 2 维矩阵，可以用 *A*(*i*, *j*)来表示第 *i* 行第 *j* 列的元素。例如，假设 *A*=magic(3),

那么 $A(3,2)$代表第 3 行第 2 列的数字。代码设置如下：

```
A=magic(3)
b=A(3,2)
```

由上述语句得到输出代码如下：

```
A =
      8     1     6
      3     5     7
      4     9     2
b =
      9
```

也可以通过下标来改变矩阵的值，代码设置如下：

```
A(3,2)=0
```

由上述语句得到新的矩阵 ***A*** 如下：

```
A =
      8     1     6
      3     5     7
      4     0     2
```

如果要访问多维矩阵可以通过指定多个下标来实现。

2．线性引用矩阵元素

在 MATLAB 7.0 中可以通过单下标来引用矩阵元素，其引用格式为 A(k)。通常这样的引用是用于行向量或列向量。但是这样的引用也可以用于二维矩阵。MATLAB 7.0 存储矩阵元素时并不是按照其命令行输出矩阵的格式来存储矩阵的。实际上，矩阵可以看成是按列优先排列的一个长列向量格式来存储的。例如下面的代码：

```
A = [2 6 9; 4 2 8; 3 0 1]
```

由上述语句得到矩阵：

```
A =
      2     6     9
      4     2     8
      3     5     1
```

矩阵 ***A*** 实际上在内存中是被存储成以 2、4、 3、 6、 2、 5、 9、 8、 1 排列的一个列向量。***A*** 矩阵的第 3 行第 2 列，也就是值为 5 的元素实际上在存储空间是第 6 个元素。要访问这个元素，你可以用 $A(3,2)$格式，也可以用 $A(6)$格式。后者也就是线性引用矩阵元素方法。

一般地，设矩阵 ***A*** 是一个 $M´N$ 的矩阵，矩阵元素 $A(i,j)$如果换成用线性引用矩阵元素方法来表示可以表示为 $A((j-1)*M+i)$。在上面的例子中 $A(3,2)=A((2-1)*3+3)=A(6)$。

3．引用矩阵元素方式转换

如果你有矩阵的下标，但是你却想用线性引用矩阵元素方法来访问矩阵，你可以用 sub2ind()函数来得到线性引用的下标。例如下面的代码：

```
A = [2 6 9; 4 2 8; 3 0 1];
linearindex = sub2ind(size(A), 3, 2)
```

由上述语句得到输出代码如下：

```
linearindex =    6
```

反之，如果想从线性引用的下标得到矩阵的下标可以用函数 ind2sub()，例如下面的代码：

```
[row col] = ind2sub(size(A), 6)
```

由上述语句得到输出代码如下：

```
row =
        3
col =
        2
```

4．访问多个矩阵元素

设 *A*=magic(4)，如果需要计算第 4 列元素的和，按照前面介绍的方法可以用下式来实现：

```
A(1,4) + A(2,4) + A(3,4) + A(4,4)
```

在下标表达式里，可以用冒号来表示矩阵的多个矩阵元素。例如，*A*(*1:k,j*)表示矩阵第 *j* 列的前 *k* 个元素。利用冒号，第 4 列元素的和可以用更为简洁的式子，代码设置如下：

```
sum(A(1:4, 4))
```

还有更简洁的方法，因为冒号本身可以表示一列或者一行的所有元素，所以上式可以表达为：

```
sum(A(:,4))
```

在 MATLAB 7.0 中提供了一个关键字 end，用于表示该维中的最后一个元素。所以上式还可以改写成：

```
sum(A(:,end))
```

实际上还可以用冒号来表示非相邻的多个元素，例如下面的代码：

```
A=1:10
B=A(1:3:10)
```

由上述语句得到输出代码如下：

```
A =
        1        2        3        4        5        6        7        8        9       10
B =
        1        4        7       10
```

2.2.4　矩阵信息的获取

本小节介绍如何获取矩阵的信息，包括矩阵的尺寸大小，矩阵元素的数据类型和矩阵的数据结构。

1．矩阵尺寸信息

矩阵尺寸函数可以得到矩阵的形状和大小信息，这些函数如表 2-6 所示。

表 2-6 矩阵尺寸函数

函数名	函数描述	基本调用格式	
length	矩阵最长方向的长度	*n* = length(*X*)	相当于 max(size(*X*))
ndims	矩阵的维数	*n* = ndims(*A*)	矩阵的维数
numel	矩阵的元素个数	*n* = numel(*A*)	矩阵的元素个数
size	矩阵在各个方向的长度	*d* = size(*X*) [*m*,*n*] = size(*X*) *m* = size(*X*,*dim*)	返回的大小信息以向量方式存储 返回的大小信息以分开存储 返回某一位的大小信息

下面例子用于表明如何使用矩阵尺寸信息获取函数。

设矩阵 ***A*** 为：

```
A=rand(3,5)*10
```

上述语句得到矩阵 ***A*** 如下：

```
A =
    4.0571    4.1027    3.5287    1.3889    6.0379
    9.3547    8.9365    8.1317    2.0277    2.7219
    9.1690    0.5789    0.0986    1.9872    1.9881
```

求矩阵 ***A*** 的元素平均值可以用 numel 函数来实现，代码设置如下：

```
mean=sum(A(:))/numel(A)
```

由上述语句得到均值如下：

```
mean =
    4.2740
```

在上式中 sum(*A*(:))表示对 *A* 的所有元素求和，函数 sum 的用法将在第 3 章中介绍。

2．矩阵元素的数据类型

获得矩阵元素的数据类型信息的函数如表 2-7 所示。

表 2-7 获得矩阵元素的数据类型的函数

函数名	函数描述	基本调用格式
class	返回输入数据的数据类型	*C* = class(*obj*)
isa	判断输入数据是否为指定数据类型	*K* = isa(*obj*,'*class_name*')
iscell	判断输入数据是否为单元型	*tf* = iscell(*A*)
iscellstr	判断输入数据是否为单元型的字符串	*tf* = iscellstr(*A*)
ischar	判断输入数据是否为字符数组	*tf* = ischar(*A*)
isfloat	判断输入数据是否为浮点数	*tf* =isfloat(*A*)
isinteger	判断输入数据是否为整数	*tf* =isinteger(*A*)
islogical	判断输入数据是否为逻辑型	*tf* = islogical(*A*)
isnumeric	判断输入数据是否为数值型	*tf* = isnumeric(*A*)
isreal	判断输入数据是否为实数	*tf* = isreal(*A*)
isstruct	判断输入数据是否为结构体	*tf* = isstruct(*A*)

例如矩阵 ***A*** 是一个具有实数和复数的矩阵，如果需要将实数和复数分开，可以使用表 2-7 中的函数来实现，其方法如下：

```
%定义一个具有实数和复数的矩阵
A=[5 6.5 2+3i 3.5 6 1+2i];
%定义存储实数和复数的矩阵目前为空矩阵
```

```
real_array=[];
complex_array=[];
for i=1:length(A),
%判断矩阵元素是否为实数
    if isreal(A(i))==1,
        real_array=[real_array A(i)];
    else
        complex_array=[complex_array A(i)];
    end;
end;
%输出实数元素
real_array
%输出复数元素
complex_array
```

由上述语句得到输出代码如下：

```
real_array =
    5.0000    6.5000    3.5000    6.0000
complex_array =
   2.0000 + 3.0000i    1.0000 + 2.0000i
```

3．矩阵的数据结构

测试矩阵是否为某一种数据结构的函数如表 2-8 所示。

表 2-8　测试矩阵数据结构的函数

函数名	函数描述	基本调用格式
isempty	测试矩阵是否为空矩阵	*tf* = isempty(*A*)
isscalar	测试矩阵是否为标量	*tf* = isscalar(*A*)
issparse	测试矩阵是否为稀疏矩阵	*tf* = issparse(*A*)
isvector	测试矩阵是否为矢量	*tf* = isvector(*A*)

2.2.5　矩阵结构的改变

可以改变矩阵结构的函数如表 2-9 所示。

表 2-9　改变矩阵结构的函数

函数名	函数描述	基本调用格式	
reshape	按照列的顺序重新排列矩阵元素	*B* = reshape(*A*,*m*,*n*)	把矩阵 ***A*** 变为 *m*´*n* 大小
rot90	旋转矩阵 90°	*B* = rot90(*A*) *B* = rot90(*A*,*k*)	旋转矩阵 90° 旋转矩阵 *k**90°，*k* 为整数
fliplr	以竖直方向为轴做镜像	*B* = fliplr(*A*)	
flipud	以水平方向为轴做镜像	*B* = flipud(*A*)	
flipdim	以指定的轴做镜像	*B* = flipdim(*A*,*dim*) *dim*=2	*dim*=1 以水平方向为轴做镜像， 以竖直方向为轴做镜像
transpose	矩阵的转秩	*B* = transpose(*A*)	相当于 *B*=*A*.'

续表

ctranspose	矩阵的共轭转秩	*B* = ctranspose(*A*)　　相当于 *B*=*A*′

下面以一些例子来说明这些函数的用法。下面例子中的矩阵 ***A*** 均如下定义：

```
A = [1 4 7 10; 2 5 8 11; 3 6 9 12]
```

即矩阵 ***A*** 为：

```
A =
     1     4     7    10
     2     5     8    11
     3     6     9    12
```

例如，当要将一个 3×4 的矩阵按照列的顺序重新排列成 2×6 的矩阵时，可以用下列语句来实现：

```
B = reshape(A, 2, 6)
```

由上述语句得到矩阵 ***B*** 为：

```
B =
     1     3     5     7     9    11
     2     4     6     8    10    12
```

例如，需要把矩阵旋转 90°，可以用下面语句来实现：

```
B = rot90(A)
```

由上述语句得到矩阵 ***B*** 为：

```
B =
    10    11    12
     7     8     9
     4     5     6
     1     2     3
```

例如，需要把矩阵以竖直方向为轴做镜像，可以用下面语句来实现：

```
B = fliplr(A)
```

由上述语句得到矩阵 ***B*** 为：

```
B =
    10     7     4     1
    11     8     5     2
    12     9     6     3
```

2.2.6　稀疏矩阵

在 MATLAB 7.0 中可以用两种方式来存储矩阵，即满矩阵存储方式和稀疏矩阵存储方式，简称满矩阵和稀疏矩阵。

在很多情况下一个矩阵只有少数的元素是非零的，而前面的章节中介绍的矩阵对于零值和非零值是花费同样空间来存储的，因此这种存储方式会浪费很多存储空间，有时还会减慢计算的速度。这种存储方式称为满矩阵存储方式。而稀疏矩阵在 MATLAB 7.0 内部是以非零元素和非零元素的行列指标数组来表示的。因此，稀疏矩阵提供了一种针对矩阵元素大多数

都是零值矩阵的有效存储方式。

用户可以创建双精度类型、复数类型和逻辑类型的稀疏矩阵，所有 MATLAB 7.0 自带的数学函数、逻辑函数和引用操作均可以使用在稀疏矩阵上。稀疏矩阵不能自动生成，定义在满矩阵的运算只能生成满矩阵，不论多少个元素为零。但是，一旦以稀疏矩阵来存储，那么稀疏矩阵的存储方式就会传播下去。即定义在稀疏矩阵上的运算生成稀疏矩阵，定义在满矩阵上的运算生成满矩阵。

1．稀疏矩阵的存储方式

MATLAB 7.0 内部使用 3 个矩阵来存储稀疏矩阵。设有一个 $m \times n$ 的矩阵有 *nnz* 个非零元素，存储在长度为 *nzmax* 的矩阵中。

- 第 1 个矩阵存储着所有的非零元素，该矩阵的长度为 *nzmax*;
- 第 2 个矩阵存储着所有的非零元素的行指标，该矩阵的长度也为 *nzmax*;
- 第 3 个矩阵存储着每一列的开始处指针和一个标志着这 3 个矩阵结束的指针，该矩阵的长度为 n+1。

一个稀疏矩阵要求存储 *nzmax* 个浮点数和 *nzmax+n+1* 个整数，因此存储一个稀疏矩阵需要 8**nzmax* + 4*(*nzmax+n+1*)个字节的单元。

复数类型的稀疏矩阵用第 4 个矩阵来存储非零元素的虚部。一个复数只要其实部和虚部的其中一个为非零元素，该复数就是非零。

2．稀疏矩阵的创建

MATLAB 7.0 提供转换函数可以从满矩阵得到稀疏矩阵，即 sparse()函数。其调用格式为 *S* = sparse(*A*)，其中 ***A*** 为满矩阵。例如下面的示例代码：

```
A = [ 0    0    1    0
      0    2    0    0
      3    0    0    0
      0    0    4    0];
S = sparse(A)
```

由上述语句得到输出代码如下：

```
S =
   (3,1)        3
   (2,2)        2
   (1,3)        1
   (4,3)        4
```

把一个稀疏矩阵转换为一个满矩阵，可以用转换函数 full()。例如下面的示例代码：

```
A = full(S)
```

由上面语句返回得到矩阵 ***A***。

还可以用 sparse()函数来直接创建稀疏矩阵，其调用格式为 *S* = sparse(*i,j,s,m,n*)，其中 *i* 和 *j* 分别是稀疏矩阵非零元素的行和列指标，*s* 为相应的非零元素的值。*m* 和 *n* 分别是矩阵的行高度和列高度。上述稀疏矩阵可以用直接方法创建，语句如下：

```
S=sparse([1,2,3,4],[3,2,1,3],[1,2,3,4],4,4)
```

由上述语句得到输出代码如下：

```
S =
   (3,1)        3
   (2,2)        2
   (1,3)        1
   (4,3)        4
```

MATLAB 7.0 还提供一些函数用于创建一些特殊的稀疏矩阵，这些函数如表 2-10 所示。

表 2-10　　特殊稀疏矩阵创建函数

函数名	函数描述	基本调用格式
speye	创建单位稀疏矩阵	*S* = speye(*m*,*n*)　创建 *m*×*n* 单位稀疏矩阵 *S* = speye(*n*)　创建 *n*×*n* 单位稀疏矩阵
spones	创建非零元素为 1 的稀疏矩阵	*R* = spones(*S*)　把矩阵 ***S*** 的非零元素的值改为 1
sprand	创建非零元素为均匀分布的随机数的稀疏矩阵	*R* = sprand(S)　把矩阵 ***S*** 的非零元素的值改为均匀分布的随机数 *R* = sprand(m,n,density)　创建非零元素密度为 *density* 的 *m*×*n* 大小的均匀分布的随机数
sprandn	创建非零元素为高斯分布的随机数的稀疏矩阵	*R* = sprandn(*S*)　把矩阵 ***S*** 的非零元素的值改为高斯分布的随机数 *R* = sprandn(*m*,*n*,*density*)　创建非零元素值密度为 *density* 的 *m*×*n* 大小的高斯分布的随机数
sprandsym	创建非零元素为高斯分布的随机数的对称稀疏矩阵	*R* = sprandsym(*S*)　返回对称随机稀疏矩阵，其下三角和主对角结构与 ***S*** 相同 *R* = sprandsym(*n*,*density*)　返回 *n*×*n* 的对称随机稀疏矩阵，其非零元素密度为 *density*
spdiags	创建对角稀疏矩阵	*A* = spdiags(*B*,*d*,*m*,*n*)　把 ***B*** 中的值放在 ***d*** 中指定的对角线上，创建一个 *m*×*n* 的稀疏矩阵
spalloc	为稀疏矩阵分配空间	*S* = spalloc(*m*,*n*,*nzmax*)　相当于 sparse([],[],[],*m*,*n*,*nzmax*)

还可以将外部的稀疏矩阵转换成为内部的形式。如果 ASCII 码文件中存有一个稀疏矩阵的[i,j,v]，即非零值的行列指标和非零值，则可以用 load()函数把该文件加载到 MATLAB 7.0 空间，然后用函数 spconvert()把它转换为稀疏矩阵。例如，文件 t.dat 的内容如下：

```
3 1   3
2 2   2
1 3   1
4 3   4
```

然后用如下语句得到稀疏矩阵：

```
load t.dat
S = spconvert(t)
```

上述语句得到输出代码如下：

```
S =
```

```
(3,1)        3
(2,2)        2
(1,3)        1
(4,3)        4
```

3．查看稀疏矩阵

MATLAB 7.0 还提供一些函数，用于得到稀疏矩阵的定量信息和图形化信息。这些函数包括得到稀疏矩阵非零值信息和图形化稀疏矩阵。

查看稀疏矩阵非零值信息的函数如表 2-11 所示。

表 2-11　　查看稀疏矩阵非零值信息的函数

函数名	函数描述	基本调用格式
nnz	返回非零值的个数	n = nnz(X)
nonzeros	返回非零值	s = nonzeros(A)
nzmax	返回用于存储非零值的空间长度	n = nzmax(S)

为了尝试上面函数的用法，可以加载 MATLAB 7.0 自带的稀疏矩阵 west0479，代码设置如下：

```
load west0479
whos west0479
```

由上述语句得到输出代码如下：

```
Name              Size                         Bytes    Class
   west0479       479x479                        24564    double array (sparse)
```

为了得到该矩阵非零值的个数，可以用如下语句：

```
nnz(west0479)
```

由上述语句得到输出代码如下：

```
ans =
         1887
```

为了得到用于存储非零值的空间长度，可以用如下语句：

```
nzmax(west0479)
```

上述语句得到输出如下：

```
ans =
         1887
```

有时用图形来显示矩阵非零值的分布是很有用的，在 MATLAB 7.0 中 spy()函数实现了这一功能。为了查看稀疏矩阵 west0479 的非零值的分布，可以用如下语句：

```
spy(west0479)
```

由上述语句得到如图 2-7 所示的图形。

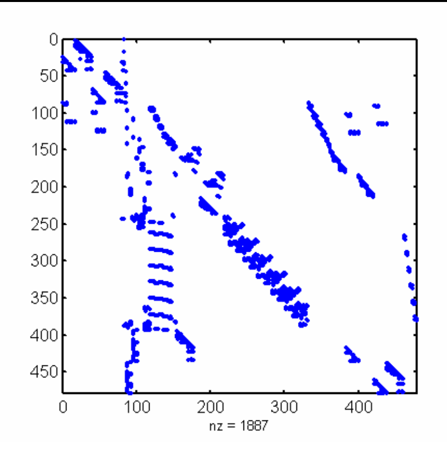


图 2-7　稀疏矩阵图

4．稀疏矩阵的运算规则

在 MATLAB 7.0 系统中的各种命令都可以用于稀疏矩阵的运算。但有稀疏矩阵参加运算时，所得到的结果将遵循以下的规则：

- 把矩阵变为标量或者定长向量的函数总是给出满矩阵;
- 把标量或者定长向量变换到矩阵的函数(zeros()、ones()、eye()、rand()等)总是给出满矩阵；而能给出稀疏矩阵结果的相应的函数有 speye()和 sprand()等;
- 从矩阵到矩阵或者向量的变换函数将以原矩阵的形式出现。即定义在稀疏矩阵上的运算生成稀疏矩阵，定义在满矩阵上的运算生成满矩阵。例如 chol(*S*)、max(*S*)和 sum(*S*)等函数;
- 两个矩阵运算符（如+、-、*、\、|）操作后的结果一般都是满矩阵，除非参加运算的矩阵都是稀疏矩阵，或者操作本身（如.*，&）保留矩阵的稀疏性;
- 参与的矩阵扩展（如[*A B*;*C D*]）的子矩阵中，只要有一个是稀疏的，那么所得的结果也是稀疏的;
- 在矩阵引用中，将仍以原矩阵形式给出结果。若 ***S*** 矩阵是稀疏的，而 ***Y*** 矩阵是全元素的，不管 *I*、*J* 是标量还是向量，那么“右引用” *Y*=*S*(*I*,*J*)产生稀疏矩阵，而“左引用” *S(I,J)*=*Y* 产生满矩阵。

2.3　运算符和特殊符号

MATLAB 7.0 中提供了丰富的运算符，非常方便地满足用户的各种应用。这些运算符包括算数运算符、关系运算符和逻辑运算符三种运算符。

2.3.1　算数运算符

MATLAB 7.0 的算数运算符的用法和功能如表 2-12 所示。

表 2-12　　　　　　　　　　　　　　算数运算符的用法和功能

运算符	用法	功能描述
+	*A*+*B* +*A*	加法或者一元运算符正号。***A*+*B*** 把矩阵 ***A*** 和 ***B*** 相加。***A*** 和 ***B*** 必须是具有相同长度的矩阵，除非它们之一为标量。标量可以与任何一个矩阵相加
-	*A*-*B* -*A*	减法或者一元运算符负号。***A*-*B*** 把矩阵 ***A*** 减去 ***B***。***A*** 和 ***B*** 必须是具有相同长度的矩阵，除非它们之一为标量。标量可以被任何一个矩阵减去
.*	*A*.**B*	元素相乘。***A*.*B*** 相当于 ***A*** 和 ***B*** 对应的元素相乘。对于非标量的矩阵 ***A*** 和 ***B***，矩阵 ***A*** 列长度必须和矩阵 ***B*** 的行长度一致。一个标量可以与任何一个矩阵相乘
./	*A*./*B*	元素的右除法。矩阵 ***A*** 除以矩阵 ***B*** 的对应元素，即等于 *A*(*i*,*j*)/*B*(*i*,*j*)。对于非标量的矩阵 ***A*** 和 ***B***，矩阵 ***A*** 列长度必须和矩阵 ***B*** 的行长度一致
.\	*A*.*B*	元素的左除法。矩阵 ***B*** 除以矩阵 ***A*** 的对应元素，即等于 *B*(*i*,j)/*A*(*i*,*j*)。对于非标量的矩阵 ***A*** 和 ***B***，矩阵 ***A*** 列长度必须和矩阵 ***B*** 的行长度一致
.^	*A*.^*B*	元素的乘方。等于[*A*(*i*,*j*)^*B*(*i*,*j*)]，对于非标量的矩阵 ***A*** 和 ***B***，矩阵 ***A*** 列长度必须和矩阵 ***B*** 的行长度一致
.’	*A*.’	矩阵转秩。当矩阵是复数时，不求矩阵的共轭
*	*A***B*	矩阵乘法。对于非标量的矩阵 ***A*** 和 ***B***，矩阵 ***A*** 列长度必须和矩阵 ***B*** 的行长度一致。一个标量可以与任何一个矩阵相乘
/	*A*/*B*	矩阵右除法。粗略地相当于 ***B****inv(***A***)，准确地说相当于(***A***'***B***')'。方程 ***X**A*=*B*** 的解
\	*A**B*	矩阵左除法。粗略地相当于 inv(***A***)****B***。方程 ***A**X*=*B*** 的解
^	*A*^*B*	矩阵乘方。具体用法参见表后面的补充说明
’	*A*’	矩阵转秩。当矩阵是复数时，求矩阵的共轭转秩

补充说明符号^的用法如下：

当 *A* 和 *B* 都是标量时，表示标量 *A* 的 *B* 次方幂。当 ***A*** 为方阵，*B* 为正整数时，表示矩阵 ***A*** 的 *B* 次乘积；*B* 为负整数时，表示矩阵 ***A*** 的逆的 *B* 次乘积。当 *B* 为非整数时，有如下表达式：

$$A\wedge B=V*\begin{bmatrix} l_1^{\mathrm{B}} & & & \\ & \cdot & & \\ & & \cdot & \\ & & & \cdot & \\ & & & & l_n^{\mathrm{B}} \end{bmatrix}/V$$

其中 l_1^{B},L ,l_n^{B} 为矩阵 ***A*** 的特征值，***V*** 为对应的特征向量矩阵。当 *A* 为标量，***B*** 为方阵时，有如下表达式：

$$A\wedge B=V*\begin{bmatrix} A^{l_1} & & & \\ & \cdot & & \\ & & \cdot & \\ & & & \cdot & \\ & & & & A^{l_n} \end{bmatrix}/V$$

其中 A^{l_1},L ,A^{l_n} 为方阵 ***B*** 的特征值，***V*** 为对应的特征向量矩阵。当矩阵 ***A*** 和矩阵 ***B*** 都为矩阵时，此运算无定义。

除了某些矩阵运算符，MATLAB 7.0 的算数运算符只对相同规模的数组作相应的运算。

对于向量和矩阵，两个操作数必须同规模或者有一个操作数为标量。如果一个操作数是标量，而另外一个不是，MATLAB 7.0 会将这个标量与另一个操作数的每一个元素进行运算。

例如：

```
A = magic(3)
```

上述语句得到矩阵如下：

```
A =
     8     1     6
     3     5     7
     4     9     2
```

在命令行输入语句如下：

```
3*A
```

上述语句得到输出代码如下：

```
ans =
    24     3    18
     9    15    21
    12    27     6
```

与以往的 MATLAB 版本不同的是，MATLAB 7.0 的数学运算符不但支持双精度数据类型的运算，还增加了对单精度类型、1 字节无符号整数、1 字节有符号整数、2 字节无符号整数、2 字节有符号整数、4 字节无符号整数和 4 字节有符号整数运算的支持。

2.3.2　关系运算符

MATLAB 7.0 的关系运算符的用法和功能如表 2-13 所示。

表 2-13　关系运算符

运算符	功能描述
<	小于
<=	小于等于
>	大于
>=	大于等于
==	等于
~=	不等于

MATLAB 7.0 的关系运算符只对具有相同规模的两个操作数或者其中一个操作数为标量的操作数进行操作。当两个操作数具有相同规模时，MATLAB 7.0 对两个矩阵的对应元素进行比较，返回的结果是与操作数具有相同规模的矩阵。例如，下面的示例代码比较 3×3 魔方矩阵中哪些元素的值大于 4。

```
magic(3)>4*ones(3)
```

上述语句得到输出代码如下：

```
ans =
     1     0     1
     0     1     1
     0     1     0
```

返回结果中等于 1 的位置上，表示此处 magic(3)的矩阵元素大于 4。

当其中一个操作数是标量时，MATLAB 7.0 将标量与另一个操作数的每一个元素进行比较，返回结果是与非标量操作数具有相同规模的矩阵。例如，上面的示例也可以用矩阵与标量进行关系运算来实现，语句如下：

```
magic(3)>4
```

当检验一个矩阵是否是空矩阵时，或许会想到用关系式 *A*==[]。但是如果 ***A*** 不是 0×0 或 1×1 的话，将返回警告信息。MATLAB 7.0 提供了一个函数 isempty()专门用于判断一个矩阵是否为空矩阵，例如要检验 ***A*** 是否为空矩阵，可以用语句 isempty(*A*)。如果 ***A*** 为空矩阵则返回 1，否则返回 0。

2.3.3　逻辑运算符

MATLAB 7.0 提供 3 种类型的逻辑运算符，即元素方式逻辑运算符、比特方式逻辑运算符和短路逻辑运算符。

元素方式逻辑运算符的用法和功能如表 2-14 所示。元素方式逻辑运算符只接受逻辑类型变量输入。表中例子采用如下矩阵：

```
A = [0 1 1 0 1];
B = [1 1 0 0 1];
```

表 2-14　　元素方式逻辑运算符

运算符	功能描述	例子
&	逻辑与。两个操作数同时为 1，运算结果为 1；否则为 0	*A*&*B*　= 01001
|	逻辑或。两个操作数同时为 0，运算结果为 0；否则为 1	*A*|*B*　= 11101
~	逻辑非。当 *A* 为 0 时，运算结果为 1；否则为 0	~*A* = 10010
xor	逻辑异或。但两个操作数相同时，运算结果为 0；否则为 1	xor(*A*,*B*)=10100

MATLAB 7.0 的元素方式逻辑运算符只对具有相同规模的两个操作数或者其中一个操作数为标量的操作数进行操作。

元素方式逻辑运算符有重载的函数，实际上符号' & '、' | '和 ' ~ '的重载函数分别是 and()、or()和 not()。

比特方式逻辑运算符对操作数的每一个比特位进行逻辑操作，其用法和功能如表 2-15 所示。比特方式逻辑运算符接受逻辑类型和非负整数变量输入。

表中例子采用如下矩阵：

```
A = 28;                    % binary 11100
B = 21;                    % binary 10101
```

表 2-15　　比特方式逻辑运算符

函数名	功能描述	例子
bitand	位与。返回两个非负整数的对应位相与操作	bitand(*A*,*B*) = 20 (binary 10100)
bitor	位或。返回两个非负整数的对应位做或操作	bitor(*A*,*B*) = 29 (binary 11101)
bitcmp	补码。返回 ***n*** 位整数表示的补码	bitcmp(*A*,5) = 3 (binary 00011)
bitxor	位异或。返回两个非负整数的对应位做异或操作	bitxor(*A*,*B*) = 9 (binary 01001)

MATLAB 7.0 的短路逻辑操作符用法和功能如表 2-16 所示。

表 2-16　　短路逻辑运算符

<table>
<tr><th>运算符</th><th>功能描述</th></tr>
<tr><td>&&</td><td>逻辑与。两个操作数同时为 1，运算结果为 1；否则为 0</td></tr>
<tr><td>||</td><td>逻辑或。两个操作数同时为 0，运算结果为 0；否则为 1</td></tr>
</table>

短路逻辑运算符的运算结果和元素方式逻辑运算符的运算结果是一样的。然而短路逻辑运算符在执行时，只有在运算结果还不确定时才去参考第二个操作数。例如，*A*&&*B* 操作，当 *A* 为 0 时，直接返回 0，而不检查 *B* 的值；当 *A* 为 1 时，如果 *B* 为 1，则返回 1，否则返回 0。*A*||*B* 的执行方式也与 *A*&&*B* 类似。

有些情况下，需要满足特定条件，才能被执行的语句，可以用短路逻辑运算符来实现。例如，下面语句实现判断是否 *a/b>1*，当 *b* 为 0 时也不会出错。当 *b*=0 时，函数直接返回 0，而不计算 a/b。

```
x = (b ~= 0) && (a/b > 1)
```

2.3.4　运算优先级

用户编写的表达式可以包括算术运算符、关系运算符和逻辑运算符。因此运算符的优先级决定了对一个表达式的求值顺序。具有相同优先级的运算符则从左到右依次进行运算；不同优先级的运算符则先进行高优先级运算。运算符的优先等级如表 2-17 所示。

表 2-17　　运算符的优先等级

<table>
<tr><th>运算符</th><th>优先等级</th></tr>
<tr><td>括号</td><td>最高优先级</td></tr>
<tr><td>转秩（.’），幂（.^），复共轭转秩（’），矩阵幂（^）</td><td></td></tr>
<tr><td>一元正号（+），一元负号（-），逻辑非（）</td><td></td></tr>
<tr><td>元素相乘（.*），元素右除（./），元素左除（.\），矩阵乘法（*），矩阵右除（/），矩阵左除（/）</td><td></td></tr>
<tr><td>加法（+），减法（-）</td><td></td></tr>
<tr><td>冒号运算符（:）</td><td></td></tr>
<tr><td>小于（<），小于等于（<=），大于（>），大于等于（>=），等于（==），不等于（~=）</td><td></td></tr>
<tr><td>逻辑与（&）</td><td></td></tr>
<tr><td>逻辑或（|）</td><td></td></tr>
<tr><td>短路逻辑与（&&）</td><td></td></tr>
<tr><td>短路逻辑或（||）</td><td>最低优先级</td></tr>
</table>

由表 2-17 中可以看到，括号的优先级别最高，因此可以用括号来改变默认的优先等级，例如下面的示例代码：

```
A = [3 9 5];
B = [2 1 5];
C = A./B.^2
```

上述语句得到输出代码如下：

```
C =
    0.7500    9.0000    0.2000
```

而采用括号后的语句如下：

```
C = (A./B).^2
```

上述语句输出结果代码如下：

```
C =
    2.2500    81.0000    1.0000
```

2.4　字符串处理函数

本节介绍字符串处理函数。字符串是指 $1\times n$ 的字符数组。

MATLAB 7.0 能够很好地支持字符串数据，用户可以用两个不同的方式来表示字符串，即字符数组和字符串单元数组。用户可以用 $m\times n$ 的字符数组来表示多个字符串，只要这些字符串的长度是一样。当需要保存多个不同长度的字符串时，可以用单元类型来实现。

MATLAB 7.0 提供了很多字符串操作，包括字符串的创建、合并、比较、查找以及与数值之间的转换。

2.4.1　字符串的构造

1．创建字符数组

可以用一对单引号来表示字符串，例如下面的示例代码：

```
str= 'I am a great person ';
```

也可以用字符串合并函数 strcat()来得到一个新的字符串，例如下面的示例代码：

```
a='My name is ';
b='Clayton Shen';
c=strcat(a,b)
```

上述语句得到输出代码如下：

```
c =
My name isClayton Shen
```

值得注意的是，函数 strcat()在合并字符串的同时会把字符串结尾的空格删除。要保留这些空格，可以用矩阵合并符[]来实现字符串合并，例如下面的示例代码：

```
a='My name is ';
b='Clayton Shen';
c=[a b]
```

上述语句得到输出代码如下：

```
c =
My name is Clayton Shen
```

用户也可以构造二维字符数组，不过要注意保持二维字符数组的每一行具有相同的长度。例如，下面字符串是合法的，因为它的每行都有 6 个字符：

```
str=['second';'string']
```

上述语句得到输出代码如下：

```
str =
second
string
```

当构造的多个字符串具有不同长度时，可以在字符串的尾部添加空格来强制实现字符串具有相同的长度。

例如下面的示例代码：

```
str=['name   ';'string']
```

上述语句得到输出代码如下：

```
str =
name
string
```

一个更简单的方法是利用函数 char()来创建字符串。该函数创建字符串数组时，如果字符串不具有相同的长度，则函数 char()自动用空格把字符串补足到最长的字符串的长度。例如下面的示例代码：

```
c=char('first','second')
```

上述语句得到输出代码如下：

```
c =
first
second
```

与函数 char()具有类似功能的函数是 strvcat()。函数 strvcat()把多个字符串合并为一个字符串数组。当字符串不具有相同长度时，函数 strvcat()自动在尾部添加空格补足到最长的字符串的长度。例如下面的示例代码：

```
c=strvcat('name','string')
```

上述语句得到输出代码如下：

```
c =
name
string
```

2．创建字符串单元数组

可以用函数 cellstr()来创建字符串单元数组。例如，有一个字符数组如下：

```
data=['name   ';'string']
```

上述语句得到输出代码如下：

```
data =
name
string
```

下面语句的功能是把上述字符数组转换成为字符串单元数组，示例代码设置如下：

```
celldata=cellstr(data)
```

上述语句得到输出代码如下：

```
celldata =
    'name'
```

```
        'string'
```

注意，函数 cellstr()已经把字符串尾部的空格截去。可以查看 *celldata* 的第一个字符串长度如下：

```
length(celldata{1})
```

上述语句得到输出代码如下：

```
ans =
     4
```

可以用函数 char()把一个字符串单元数组转换成一个字符数组，示例代码设置如下：

```
chararray=char(celldata)
```

上述语句得到输出代码如下：

```
chararray =
name
string
```

其第一个字符串的长度可以得到如下的代码：

```
length(chararray(1,:))
```

上述语句得到输出代码如下：

```
ans =
     6
```

2.4.2　字符串比较函数

MATLAB 7.0 里包括以下几种比较字符串和字符子串的方法。比较两个字符串或者两个字符串的子串是否相同。

- 比较两个字符串中的单独字符是否相同;
- 对字符串内的元素进行识别，判定每一个元素是字符还是空白符（包括空格、制表符 Tab 和换行符）。

这些函数对字符数组和字符串数组都适用。

1．字符串比较函数

MATLAB 7.0 为用户提供的字符串比较函数如表 2-18 所示。

表 2-18　字符串比较函数

函数名	功能描述	基本调用格式
strcmp	比较两个字符串是否相等	strcmp(*S*1,*S*2)　如果字符串相等则返回 1，否则返回 0
strncmp	比较两个字符串的前 *n* 个字符是否相等	strncmp(*S*1,*S*2,*N*)　如果字符串的前 *N* 个字符相等则返回 1，否则返回 0
strcmpi	与 strcmp 函数功能相同，只是忽略字符串的大小写	strcmpi(*S*1,*S*2)　如果字符串相等则返回 1，否则返回 0(忽略大小写)
d strncmpi	与 strncmp 函数功能相同，只是忽略字符串的大小写	strncmpi(*S*1,*S*2,*N*)　如果字符串的前 *N* 个字符相等则返回 1，否则返回 0（忽略大小写）

例如，有两个字符串如下：

```
str1='blink';
```

```
str2='bliss';
```

由于这两个字符串不相同，故调用函数 strsmp()的结果为 0，语句如下：

```
c=strcmp(str1,str2)
```

上述语句得到输出代码如下：

```
c =
     0
```

这两个字符串的前 3 个字符是相同的，故用函数 strncmp()比较它们的前 3 个字符，会返回 1，代码设置如下：

```
c=strncmp(str1,str2,3)
```

上述语句得到输出代码如下：

```
c =
     1
```

2．用关系运算符比较字符串

可以对字符数组运用 MATLAB 7.0 的关系运算符，但是要求比较的字符数组具有相同的维数，或者是其中一个是标量。例如，可以用等号运算符（==）来判断两个字符串里哪些字符是相同的。

```
str1='carnal';
str2='casual';
c=str1==str2
```

上述语句得到输出代码如下：

```
c =
     1     1     0     0     1     1
```

当然，还可以用其他关系运算符（>、>=、<、<=、==、!=）来比较两个字符串。

2.4.3 字符串查找和替换函数

MATLAB 7.0 为用户提供的一般字符串查找和替换函数如表 2-19 所示。

表 2-19 字符串查找和替换函数

函数名	功能描述	基本调用格式
strrep	字符串替换	*str* = strrep(*str*1, *str*2, *str*3) 把 *str*1 中的 *str*2 子串替换成 *str*3
findstr	字符串内查找（两个输入对等）	*k* = findstr(*str*1,*str*2) 查找输入中较长的字符串中较短字符串的位置
strfind	字符串内查找	*k* = strfind(*str, pattern*) 查找 *str* 中 *pattern* 出现的位置 *k* = strfind(*cellstr, pattern*) 查找单元字符串 *cellstr* 中 *pattern* 出现的位置
strtok	获得第一个分隔符之前的字符串	*token* = strtok('str') 以空格符（包括空格、制表符（Tab）和换行符）为分隔符 *token* = strtok('str', *delimiter*) 输入 *delimiter* 为指定的分隔符 [*token, rem*] = strtok(...) 返回值 *rem* 为第一个分隔符之后的字符串

续表

函数名	功能描述	基本调用格式
strmatch	在字符串数组中匹配指定字符串	*x* = strmatch('str', *STRS*)　　在字符串数组 *STRS* 中匹配字符串 str，返回匹配上的字符串所在行的指标 *x* = strmatch('str', *STRS*, 'exact')　在字符串数组 *STRS* 中精确匹配字符串 str，返回匹配上的字符串的所在行指标。只有完全匹配上时，才返回字符串的行指标

下面例子是实现字符串替换：

```
s1 = 'This is a good example.';
str = strrep(s1, 'good', 'great')
```

上述语句得到输出代码如下：

```
str =
This is a great example.
```

下面例子是用于实现字符串查找：

```
str = 'This is a good example.';
index = strfind(str, 'a')
```

上述语句得到输出代码如下：

```
index =
        9       18
```

下面例子是用于获得第一个分隔符之前的字符串：

```
s = '    This is a simple example.';
[token, rem] = strtok(s)
```

上述语句得到输出代码如下：

```
token =
    This
rem =
    is a simple example.
```

下面例子是用于实现字符串匹配：

```
x = strmatch('max', strvcat('max', 'minimax', 'maximum'))
```

上述语句得到输出代码如下：

```
x =
        1
        3
```

2.4.4　字符串——数值转换

MATLAB 7.0 为用户提供的把数值转换为字符串的函数如表 2-20 所示。

表 2-20　　数值转换为字符串的函数

函数名	功能描述	例子
char	把一个数值截取小数部分，然后转换为等值的字符	[72 105]　→　'Hi'
int2str	把一个数值的小数部分四舍五入，然后转换为字符串	[72 105]　→　'72 105'

续表

函数名	功能描述	例子
num2str	把一个数值类型的数据转换为字符串	[72 105]' → 72/105/'（输出格式为%1d/）
mat2str	把一个数值类型的数据转换为字符串，返回的结果是 MATLAB 7.0 能识别的格式	[72 105]' → [72 105]'
dec2hex	把一个正整数转换为十六进制的字符串表示	[72 105] → '48 69'
dec2bin	把一个正整数转换为二进制的字符串表示	[72 105] → '1001000 1101001'
dec2base	把一个正整数转换为任意进制的字符串表示	[72 105] → '110 151'（八进制）

MATLAB 7.0 为用户提供的把字符串转换为数值的函数如表 2-21 所示。

表 2-21　　字符串转换为数值的函数

函数名	功能描述	例子
uintN	把字符转换为等值的整数	'Hi' → [72 105]
str2num	把一个字符串转换为数值类型	'72 105' → [72 105]
str2double	与 str2num 相似，但比 str2num 性能优越，同时提供对单元字符数组的支持	{'72' '105'} → [72 105]
hex2num	把一个 IEEE 格式的十六进制字符串转换为数值类型	'400921fb54442d18' → 'pi'
hex2dec	把一个 IEEE 格式的十六进制字符串转换为整数	'12B' → 299
bin2dec	把一个二进制字符串转换为十进制整数	' 010111 ' → 23
base2dec	把一个任意进制的字符串转换为十进制整数	'12' → 10（八进制）

例如，要在命令行中输出一行字符串来显示向量 *x* 的最大值，可以用如下程序：

```
x=rand(1,10);
disp(['向量 x 中的最大值为:' num2str((max(x)))]);
```

其中函数 disp()是在命令行中显示一个字符串。由上述语句得到的输出代码如下：

```
向量 x 中的最大值为:0.93547
```

第 3 章　数学运算

本章将介绍 MATLAB 7.0 中与数学运算有关的函数和概念。在 MATLAB 7.0 中一切数据均以矩阵的形式出现，它的数学运算则包括两种：一种是针对整个矩阵的数学运算，称之为矩阵运算，例如求矩阵的行列式的函数 det()；另一种是针对矩阵的每一个元素进行运算的函数，称之为矩阵元素的运算，例如求矩阵每一元素的正弦值的函数 sin()。

本章 3.1 小节中将介绍矩阵运算函数，包括矩阵分析、线性方程组、矩阵分解、矩阵的特征值和特征向量，以及非线性矩阵运算。3.2 小节将介绍矩阵元素的运算函数，包括三角函数、指数对数函数、复数函数和截断求余函数。3.3 小节介绍特殊数学函数，这些函数应该属于矩阵元素的运算，特殊数学函数包括特殊函数、数论函数和坐标变换函数。由于特殊数学函数在使用和应用场合与 3.2 小节介绍的函数差异较大，故单独做一小节介绍。

3.1　矩阵运算

矩阵运算是线性代数中极其重要的部分。MATLAB 7.0 可以支持很多线性代数中定义的操作。MATLAB 7.0 强大的矩阵运算能力使其成为优秀数值计算软件。

本小节将介绍与矩阵运算相关的内容，包括矩阵分析、线性方程组求解、特征值求解和奇异值等。

3.1.1　矩阵分析

MATLAB 7.0 提供的矩阵分析函数如表 3-1 所示。

表 3-1　　矩阵分析函数

函数名	功能描述
norm	求矩阵或者向量的范数
normest	估计矩阵的 2 阶范数
rank	矩阵的秩，即求对角元素的和
det	矩阵的行列式
trace	矩阵的迹
null	0 空间
orth	正交化空间
rref	约化行阶梯形式
subspace	求两个矩阵空间的角度

1．向量和矩阵的范数运算

对于线性空间中的一个向量 $x=\{x_1,x_2,...x_n\}$，如果存在一个函数 $\rho(x)$ 满足以下 3 个条件：

（1） $\rho(x)>0$，且 $\rho(x)=0$ 的充要条件为 $x=0$;

（2） $\rho(ax)=|a|\rho(x)$，其中 a 为任意标量；

（3）对向量 x 和 y，有 $\rho(x+y)\le\rho(x)+\rho(y)$。

称 $\rho(x)$ 为向量 x 的范数，一般记为 $\|x\|$。范数的形式多种多样，下面式子中定义的范数操作就满足以上 3 个条件：

$$\|x\|_p=\left(\sum_{i=1}^{n}|x_i|^p\right)^{1/p},\ p=1,2,\text{L}\ ,\text{且}\|x\|_\infty=\max_{1\le i\le n}|x_i|;\|x\|_{-\infty}=\min_{1\le i\le n}|x_i|$$

上式定义的 $\|x\|_p$ 称为 p 阶范数，其中最有用的是 1、2 和 ∞ 阶范数。

矩阵的范数是基于向量的范数定义的，其定义式如下：

$$\|A\|=\max_{x\ne0}\frac{\|Ax\|}{\|x\|}$$

与向量的范数一样，矩阵的范数最常用的也是 1、2 和 ∞ 阶范数。它们的定义如下：

$$\|A\|_1=\max_{1\le j\le n}\sum_{i=1}^{n}|a_{ij}|\text{，}\|A\|_2=\sqrt{S_{\max}\{A^{\mathrm{T}}A\}}\text{和}\|A\|_\infty=\max_{1\le i\le n}\sum_{j=1}^{n}|a_{ij}|$$

在上式中 $S(\boldsymbol{A})$为矩阵 $\boldsymbol{A}$ 的特征值，而 $S_{\max}\{A^{\mathrm{T}}A\}$ 则为 $\boldsymbol{A}$ 矩阵的最大奇异值的平方。

在 MATLAB 7.0 中，求向量范数的函数具体用法如下：

- N=norm(x,p)　对任意大于 1 的 p 值，返回向量 x 的 p 阶范数；
- N=norm(x)　　返回向量的 2 阶范数，相当于 N=norm(x,2)；
- N=norm(x,inf)　返回向量的 ∞ 阶范数，相当于 N=max(abs(x))；
- N=norm(x,-inf)　返回向量的-∞ 阶范数，相当于 N=min(abs(x))。

在 MATLAB 7.0 中，求矩阵范数的函数具体用法如下：

- N=norm(A)　　　计算矩阵的 2 阶范数，也就是最大奇异值；
- N=norm(A,p)　　根据参数 p 的值不同，求不同阶的范数值。当 p=1 时，计算矩阵 $\boldsymbol{A}$ 的 1 阶范数，相当于 max(sum(abs(**A**)))。当 p=2 时，计算矩阵 $\boldsymbol{A}$ 的 2 阶范数，相当于 norm(**A**)。p=inf 时，计算矩阵 $\boldsymbol{A}$ 的 ∞ 阶范数，相当于 max(sum(abs(**A'**)))。当 p='pro'时，计算矩阵 $\boldsymbol{A}$ 的 F 范数(Frobenius 范数)，相当于 sqrt(sum(diag(**A'*A**)))。

例如求向量 x 的 2 阶范数，具体代码如下：

```
norm(1:5,2)
```

由上述语句得到具体代码如下：

```
ans =
      7.4162
```

例如求单位矩阵的 1 阶范数，具体代码如下：

```
norm(eye(3),1)
```

由上述语句得到具体代码如下：

```
ans =
     1
```

当矩阵维数比较大时会导致计算矩阵范数的时间比较长，并且当一个近似的范数值满足要求时，可以考虑使用函数 normest()来估计 2 阶范数值。函数 normest()最初开发时是为了提供给稀疏矩阵使用的，同时它也能接受满矩阵的输入，一般在满矩阵维数比较大时使用。

函数 normest()的用法如下：

- normest(*S*)　估计矩阵 *S* 的 2 阶范数值，默认允许的相对误差为 1e-6；
- normest(*S*,*tol*)　使用 *tol* 作为允许的相对误差。

例如，首先用函数生成一个 1000×1000 的柯西矩阵 *W*，该柯西矩阵满足条件 $W_{ij}=1/(x_i+y_j)$，其中 x_i !=x_j, y_i !=y_j（当 i !=j），然后对该矩阵 *W* 求其 2 阶范数，最后对比函数 norm()和 normest()的效果，具体代码如下：

```
W = gallery('cauchy',1000);              %产生一个 1000×1000 的柯西矩阵
t1=clock;                                %获得系统时间
W_norm=norm(W);                          %计算范数
t2=clock;                                %获得系统时间
t_norm=etime(t2,t1)                      %计算范数的耗时
t3=clock;                                %获得系统时间
W_normest=normest(W);                    %估计范数
t4=clock;                                %获得系统时间
t_normest=etime(t4,t3)                   %估计范数的耗时
```

由上述语句得到如下代码：

```
W_norm =
    2.2469
t_norm =
    5.7660
W_normest =
    2.2469
t_normest =
    0.1410
```

由上述示例可以看出，函数 norm()计算出的范数精确值为 2.2469，耗时 5.7660s，而函数 normest()计算出的范数值也为 2.2469，但耗时为 0.1410s。这就说明了，采用函数 normest()计算范数可以大大提高计算速度。

2．矩阵的秩

矩阵 *A* 中线性无关的列向量个数称为列秩，线性无关的行向量个数称为行秩。可以证明矩阵的列秩与行秩是相等的。矩阵求秩的方法很多，其中有些算法是稳定的，有些算法是不稳定的。MATLAB 7.0 中用函数 rank()来计算矩阵的秩，所采用的算法是基于矩阵奇异值分

解基础上的，具体代码如下：

```
s = svd(A);
tol = max(size(A))*eps(max(s));
r = sum(s > tol)
```

这种算法是最耗时，同时也是最稳定的。

函数 rank()的用法如下：

- rank(***A***)　　用默认允许误差计算矩阵的秩；
- rank(***A***,*tol*)　　给定允许误差计算矩阵的秩，*tol* = max(size(***A***)) * eps(norm(***A***))。

例如，求 6 阶单位矩阵的秩，具体代码如下：

```
rank(eye(6))
```

由上述语句得到如下代码：

```
ans =
     6
```

3．矩阵的行列式

矩阵 ***A***={a_{ij}}_{n×n} 的行列式定义如下：

$|A| = \det(A) = \sum_k (-1)^k a_{1k_1} a_{2k_2} \cdots a_{nk_n}$，其中 $k_1, k_2, \cdots, k_n$ 是将序列 $1, 2, \cdots, n$ 交换 k 次所得的序列。

在 MATLAB 7.0 中用函数 det()来计算矩阵的行列式。

例如，计算矩阵 ***A***=[1 2 3;4 5 6;7 8 9]的行列式，如下：

```
A=[1 2 3;4 5 6;7 8 9];
A_det=det(A);
disp(['A 的行列式值=',num2str(A_det)]);
```

由上述语句得到如下代码：

```
A 的行列式值=0
```

矩阵 ***A*** 的行列式恰好为 0。

线性代数中定义行列式为 0 的矩阵为奇异矩阵。但是一般不能使用语句 abs(det(***A***))<=ε 来判断矩阵 ***A*** 的奇异性，因为不容易选择合适的允许误差ε来满足要求。MATLAB 7.0 提供函数 cond()可以用来判定矩阵的奇异性。

4．矩阵的迹

矩阵的迹定义为矩阵对角元素之和。在 MATLAB 7.0 中用函数 trace()来计算矩阵的迹。

例如，求矩阵 ***A***=[1 2 3;4 5 6;7 8 9]的迹，如下：

```
A=[1 2 3;4 5 6;7 8 9];
A_trace=trace(A);
disp(['A 的迹 = ',num2str(A_trace)]);
```

由上述语句得到代码如下：

```
A 的迹 = 15
```

5．矩阵的化零矩阵

如果对于非满秩矩阵 ***A***，若有矩阵 ***Z*** 使得 ***A****Z*** 的元素都为 0，且矩阵 ***Z*** 为一个正交矩阵（***Z***'****Z***=***I***），则称矩阵 ***Z*** 为矩阵 ***A*** 的化零矩阵。MATLAB 7.0 中提供了求化零矩阵的函数 null()，用法如下：

- ***Z*** = null(***A***)　　返回矩阵 ***A*** 的一个化零矩阵，如果化零矩阵不存在则返回空矩阵；
- ***Z*** = null(***A***,'r')　　返回有理数形式的化零矩阵。

例如，求矩阵 ***A***=[1 2 3;1 2 3;4 5 6]的化零矩阵，具体代码设置如下：

```
A=[1 2 3;1 2 3;4 5 6];
Z=null(A)
AZ=A*Z
```

由上述语句得到代码如下：

```
Z =
   -0.4082
    0.8165
   -0.4082
AZ =
  1.0e-015 *
    0.2220
    0.2220
   -0.8882
```

如果要求矩阵 ***A*** 的有理数形式的化零矩阵，具体代码设置如下：

```
ZR = null(A,'r')
AZR=A*ZR
```

由上述语句得到代码如下：

```
ZR =
     1
    -2
     1
AZR =
     0
     0
     0
```

6．矩阵的正交空间

矩阵 ***A*** 的正交空间 ***Q*** 具有 ***Q***'****Q***=***I*** 的性质，并且 ***Q*** 的列矢量构成的线性空间与矩阵 ***A*** 的列矢量构成的线性空间相同，且正交空间 ***Q*** 与矩阵 ***A*** 具有相同的秩。MATLAB 7.0 中提供了函数 orth()来求正交空间 ***Q***。

- ***Q*** = orth(***A***)，返回矩阵 ***A*** 的正交空间 ***Q***

例如，求出矩阵 ***A*** 的正交空间 ***Q***，并比较 ***Q*** 和 ***A*** 的秩，具体代码如下：

```
A=[1 2 3;4 5 6;7 8 9;10 11 12];     %矩阵 A
Q=orth(A)                           %正交矩阵 Q
t=(rank(A)==rank(Q))                %比较矩阵 A 与 Q 的秩
```

由上述语句得到代码如下输出：

```
Q =
    -0.1409     0.8247
    -0.3439     0.4263
    -0.5470     0.0278
    -0.7501    -0.3706
t =
     1
```

7．矩阵的约化行阶梯形式

矩阵的约化行阶梯形式是高斯-约旦消去法解线性方程组的结果，其形式为：

$$\begin{pmatrix} 1 & K & 0 & * \\ M & O & M & * \\ 0 & L & 1 & * \end{pmatrix}$$

MATLAB 7.0 中提供了函数 rref()来求矩阵的约化行阶梯形式，其用法如下：

- R = rref(***A***)，返回矩阵 ***A*** 的约化行阶梯形式 R。
- [R,jb] = rref(***A***)，返回矩阵 ***A*** 的约化行阶梯形式 R，同时返回 $1\times r$ 的向量 jb 使 r 为矩阵 ***A*** 的秩；***A***(:,jb)是矩阵 ***A*** 的列矢量构成的线性空间；R(1:r,jb)是 $r\times r$ 的单位矩阵。
- [R,jb] = rref(***A***,tol)，以 tol 作为误差容限计算矩阵 A 的秩。

例如，求出矩阵 A 的约化行阶梯形式，并比较 Q 和 A 的秩，具体代码设置如下：

```
A=[1 2 3;4 5 6;7 8 9;10 11 12];     %矩阵 A
R=rref(A)                           %正交矩阵 A 的约化行阶梯形式
t=(rank(A)==rank(R))                %比较矩阵 A 与矩阵 R 的秩
```

由上述语句得到如下代码：

```
R =
     1     0    -1
     0     1     2
     0     0     0
     0     0     0
t =
     1
```

8．矩阵空间之间的夹角

矩阵空间之间的夹角代表两个矩阵线性相关的程度，如果夹角很小，它们之间线性相关度就很高，反之，它们之间线性相关度就不大。在 MATLAB 7.0 中用函数 subspace()来实现

求矩阵空间之间的夹角，其调用格式如下。

- *theta* = subspace(*A*,*B*)，返回矩阵 ***A*** 和矩阵 ***B*** 之间的夹角。

例如，求矩阵 ***A*** 和 ***B*** 之间的夹角，代码如下：

```
A=[1 2 3;4 5 6;7 8 9;10 11 12];      %矩阵 A
B=magic(4);
subspace(A,B)
```

上述语句得到结果如下：

```
ans =
     0.6435
```

该结果表明矩阵 ***A*** 和矩阵 ***B*** 的线性相关度很低。

3.1.2　线性方程组

1．线性方程组的问题

在工程计算中，一个重要的问题是线性方程组的求解。在矩阵表示法中，上述问题可以表述为给定两个矩阵 ***A*** 和 ***B***，是否存在惟一的解 *X* 使得

AX=B 或 XA=B。

例如，求解方程 3*x*=6 就可以将矩阵 ***A*** 和 ***B*** 看成是标量的一种情况，最后得出该方程的解为 *x*=6/3=2。

尽管在标准的数学中并没有矩阵除法的概念，但 MATLAB 7.0 采用了与解标量方程中类似的约定，用除号来表示求解线性方程的解。MATLAB 7.0 采用第 2 章介绍过的运算符斜杠'/'和运算符反斜杠'\'来表示求线性方程的解，其具体含义如下：

- *X*=*A**B*　　　表示求矩阵方程 *AX*=*B* 的解；
- *X*=*B*/*A*　　　表示求矩阵方程 *XA*=*B* 的解。

对于 *X*=*A**B*，要求矩阵 ***A*** 和矩阵 ***B*** 有相同的行数，***X*** 和 ***B*** 有相同的列数，***X*** 的行数等于矩阵 ***A*** 的列数。*X*=*B**A* 行和列的性质则与之相反。

在实际情况中，形式 *AX*=*B* 的线性方程组比形式 *XA*=*B* 的线性方程组要常见得多。因此反斜杠'\'用得更多。本小节的内容也主要针对反斜杠'\'除法进行介绍。斜杠'/'除法的性质可以由恒等变换式得到(*B*/*A*)'=(*A*'*B*')。

系数矩阵 ***A*** 不一定要求是方阵，矩阵 ***A*** 可以是 $m \times n$ 的矩阵，有如下 3 种情况：

- $m=n$　恰定方程组，MATLAB7.0 会寻求精确解；
- $m>n$　超定方程组，MATLAB7.0 会寻求最小二乘解；
- $m<n$　欠定方程组，MATLAB7.0 会寻求基本解，该解最多有 *m* 个非零元素。

值得注意的是用 MATLAB 7.0 求解这种问题时，并不采用计算矩阵的逆的方法。针对不同的情况，MATLAB7.0 会采用不同的算法来解线性方程组。

2．线性方程组的一般解

线性方程组 *AX*=*B* 的一般解给出了满足 *AX*=*B* 的所有解。线性方程组的一般解可以通过下面的步骤得到。

- 解相应的齐次方程组 $AX=0$，求得基础解。可以使用函数 null()来得到基础解。语句 null(A)返回齐次方程组 $AX=0$ 的一个基础解，其他基础解与 null(A)是线性关系。
- 求非齐次线性方程组 $AX=B$，得到一个特殊解。
- 非齐次线性方程组 $AX=B$ 的一般解等于基础解的线性组合加上特殊解。

在后面的章节将介绍求非齐次线性方程组 $AX=B$ 特殊解的方法。

3．恰定方程组的求解

恰定方程组是方程的个数与未知量个数相同的方程组。恰定方程组中矩阵 **A** 是一个方阵，矩阵 **B** 可能是一个方阵或者是一个列向量。

如果恰定方程组是非奇异的，则语句 $A \backslash B$ 给出了恰定方程组的精确解，该精确解与矩阵 **B** 的维数大小一样。具体代码设置如下：

```
A=magic(3)
B=[1;2;3]
X=A\B
```

由上述语句得到如下代码：

```
X =
      0.0500
      0.3000
      0.0500
```

可以验证 $A*X$ 精确地等于矩阵 **B**。

如果恰定方程组是奇异的，则该方程的解不存在或者不惟一。执行语句 $A \backslash B$ 时，如果发现 **A** 是接近奇异的，MATLAB 7.0 将给出警告信息；如果发现 **A** 是严格奇异的，MATLAB 7.0 一方面给出警告信息，另一方面给出结果为 inf。

一个奇异的恰定方程组如果解存在，则可以用矩阵 **A** 的伪逆矩阵 pinv(A)来得到方程的一个解，其解为 pinv(A)*B。例如有矩阵 A = [1 3 7;-1 4 4;1 10 18]，其行列式为 0，可以通过如下代码验证：

```
A = [1 3 7;-1 4 4;1 10 18];
A_det=det(A);
disp(['A 的行列式=' num2str(A_det)]);
```

由上述语句得到如下代码：

```
A 的行列式=0
```

下面求解 $AX=B$，其中 B=[5;2;12]，具体代码如下：

```
B=[5;2;12];
X=pinv(A)*B
```

由上述语句得到如下代码：

```
X =
      0.3850
     -0.1103
      0.7066
```

下面计算 $A*X$，具体代码如下：

```
A*X
```

由上述语句得到如下代码：

```
ans =
    5.0000
    2.0000
   12.0000
```

可以验证 ***A****X* 精确等于 ***B***。

如果一个奇异的恰定方程组的精确解不存在，则式子 pinv(***A***)*B 返回的是均方根值最小的解。例如 ***B***=[3;6;0]，这时方程组无精确解，***A****X* 不等于 ***B***。若执行如下语句：

```
B=[3;6;0];
A*pinv(A)*B
```

由上述语句得到如下代码：

```
ans =
   -1.0000
    4.0000
    2.0000
```

可以发现，*A***X* 并不等于 *B*。

4．超定线性方程组的求解

超定线性方程组是方程的个数大于未知量个数的方程组。当进行实验数据拟合时，经常会碰到解超定线性方程组问题。下面举一个例子，设有两组如表 3-2 所示的观测数据 *x* 和 *y*。若希望采用 $y=c1*x+c2*x^2$ 的模型来拟合这两组数据。该问题中共有 11 个方程，2 个未知数，必须用最小二乘法来拟合来求解 *c*1 和 *c*2。

表 3-2　　观测数据

x	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y	-0.01	0.045	0.12	0.2	0.33	0.52	0.67	0.95	1.20	1.45	1.78

下面把该线性方程组问题用矩阵形式来表达，从而方便 MATLAB 7.0 计算。首先把 *x* 和 *y* 用列向量来表示，具体代码如下：

```
x=(0:0.1:1)';
y=([-0.01 0.045 0.12 0.2 0.33 0.52 0.67 0.95 1.2 1.45 1.78])';
```

然后构造系数矩阵 ***A***，具体代码如下：

```
A(:,1)=x';
A(:,2)=x'.^2;
```

此时方程组可以写成：*A**[c1 *c*2]'=*y*，然后用反斜杠\来求系数 *c*1 和 *c*2，具体代码如下：

```
c=A\y
```

由上述语句得到如下代码：

```
c =
    0.2420
    1.5407
```

最后，拟合得到 $y=0.242*x+1.5407*x^2$。具体代码如下：

```
y_fit=c(1)*x+c(2)*x.^2;
plot(x,y_fit,'-',x,y,'o')
```

由上述语句得到如图 3-1 所示的拟合结果和原始数据的对比图。

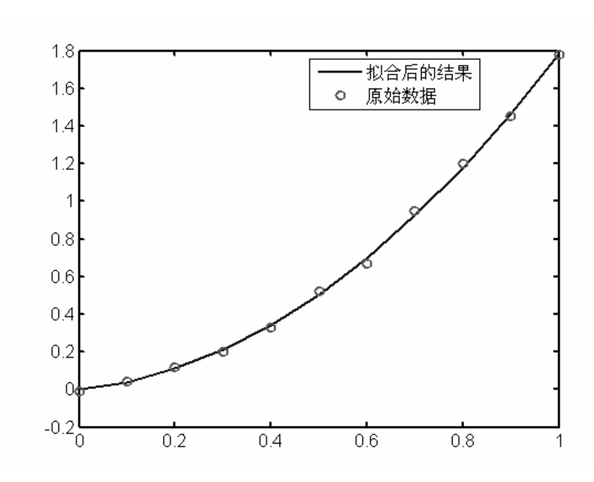


图 3-1　拟合结果和原始数据对比

由图 3-1 可见，虽然拟合得到的结果与原始数据并不严格重合，但其差别比原始数据的测量误差要小。

5．欠定线性方程组的求解

欠定线性方程组是方程组的未知量个数多于方程个数的问题。这类问题的解不是惟一的。MATLAB 7.0 将寻求一个基本解，该解至多有 m 个非零元素，然而，这些解往往也不是惟一的，MATLAB 7.0 采用的是列主元 QR 分解算法来得到特解的。

例如，求一个欠定线性方程组的解，具体代码如下：

```
A=[3 1 4 8;4 2 6 0;4 2 1 6];
B=[1;2;3];
format rat;
X=A\B
```

由上述语句得到如下代码：

```
X =
        35/37
          0
       -11/37
        -3/37
```

因为本例中矩阵大小为 3×4，所以解中共有 3 个非零元素。要得到该方程的通解还必须求方程的基本解，具体代码如下：

```
Z=null(A,'r')
```

由上述语句得到如下代码：

```
Z =
       -46/5
```

```
        74/5
         6/5
         1
```

该方程的通解可以表示为 $X_{通解} = X + Z*q$，其中 q 为任意向量，表示可以对基本解进行线性组合。

3.1.3　矩阵分解

矩阵分解是把一个矩阵分解成几个“较简单”的矩阵连乘积的形式。无论在理论上还是工程应用上，矩阵分解都是十分重要的。本节将介绍几种矩阵分解的方法，相关函数如表 3-3 所示。

表 3-3　　　　矩阵分解函数

函数	功能描述
chol	Cholesky 分解
cholinc	稀疏矩阵的不完全 Cholesky 分解
lu	矩阵 LU 分解
luinc	稀疏矩阵的不完全 LU 分解
qr	正交三角分解
svd	奇异值分解
gsvd	一般奇异值分解
schur	舒尔分解

在 MATLAB 7.0 中，线性方程组的求解主要基于 3 种基本的矩阵分解，即对称正定矩阵的 Cholesky 分解、一般方阵的高斯消去法和矩形矩阵的正交分解。这 3 种分解分别通过函数 chol()、lu()和 qr()实现。这 3 种分解都使用了三角矩阵的概念。若矩阵的所有对角线以下的元素为 0，则称为上三角矩阵；若矩阵的所有对角线以上的元素为 0，则称为下三角矩阵。

本小节将具体介绍表 3-3 列出的 5 种分解方法。

1．对称正定矩阵的 Cholesky 分解

Cholesky 分解是把一个对称正定矩阵 $\boldsymbol{A}$ 表示为一个上三角矩阵 $\boldsymbol{R}$ 与其转置的乘积，示例如下：

$$A=R'*R$$

然而，并不是所有的对称矩阵都可以进行 Cholesky 分解。能进行 Cholesky 分解的矩阵必须是正定的，即矩阵的所有对角元素必须是正的，同时矩阵的非对角元素不会太大。矩阵 i=pascal(4)就是满足以上条件的矩阵，其值为：

```
A =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
```

Cholesky 分解在 MATLAB 7.0 中用函数 chol()来实现，其调用方式如下。

- R = chol(X)，其中 $\boldsymbol{X}$ 为对称正定矩阵，$\boldsymbol{R}$ 是上三角矩阵，使得 $X=R'*R$。如果 $\boldsymbol{X}$ 是非正

定的，结果将返回出错信息。

- [R,p] = chol(X)，返回两个参数，并且不会返回出错信息。当 X 是正定矩阵时，返回的上三角矩阵 R 满足 X=R'*R，且 p=0；当 X 是非正定矩阵时，返回值 p 是正整数，R 是上三角矩阵，其阶数为 p-1，且满足 X(1:p-1,1:p-1)=R'*R。

考虑线性方程组 AX=B，其中 A 可以做 Cholesky 分解，使得 A=R'*R，这样线性方程组就可以改写成 R'*R*X=B，由于左除算符'\'可以快速处理三角矩阵，因此得出：

$$X=R\backslash(R'\backslash B)$$

如果 A 是 $n\times n$ 的方阵，chol(A)的计算复杂度是 $O(n^3)$，而左除算符'\'的计算复杂度只有 $O(n^2)$。

例如，分解 pascal(4)矩阵，具体代码如下：

```
R=chol(A)
```

由上述语句得到如下代码：

```
R =
        1      1      1      1
        0      1      2      3
        0      0      1      3
        0      0      0      1
```

对于稀疏矩阵，MATLAB 7.0 提供函数 cholinc()来做不完全 Cholesky 分解。函数 cholinc()的另一个优点是它可用于计算实半正定矩阵。函数 cholinc()的调用格式如下。

- R = cholinc(X,DROPTOL)，其中 X 和 R 的含义与函数 chol()中的变量相同，DROPTOL 为不完全 Cholesky 分解的丢失容限。当 DROPTOL 设为 0 时，退化为完全 Cholesky 分解。
- R = cholinc(X,OPTS)，其中 OPTS 为结构体，它有 3 个属性，即 DROPTOL、MICHOL 和 RDIAG。DROPTOL 为不完全 Cholesky 分解的丢失容限；MICHOL 为 1 时采用改进算法的不完全 Cholesky 分解，否则不采用改进算法；RDIAG 为 1 时 R 的对角元素中的零值替换成为 DROPTOL 的平方根，RDIAG 为 0 时不做此替换。
- R = cholinc(X,'0')，完全 Cholesky 分解。
- [R,p] = cholinc(X,'0')，返回两个参数，并且不会返回出错信息。当 X 是正定矩阵时，返回的上三角矩阵 R 满足 X=R'*R，且 p=0；当 X 是非正定矩阵时，返回值 p 是正整数，R 是上三角矩阵，其阶数为 p-1，且满足 X(1:p-1,1:p-1)=R'*R。
- R = cholinc(X,'inf')，采用 Cholesky-Infinity 分解。Cholesky-Infinity 分解是基于 Cholesky 分解的，但它可以处理实半正定矩阵。

例如，对非正定矩阵进行 chol 分解，具体代码如下：

```
S = [ 1      0      3      0;
      0     25      0     30;
      3      0      9      0;
      0     30      0    661 ];
R = chol(S);
```

由上述语句得到错误信息如下：

```
??? Error using ==> chol
Matrix must be positive definite.
```

但如果采用 Cholesky-Infinity 分解则不会出错，具体代码如下：

```
Rinf = cholinc(sparse(S),'inf');
Rinf=full(Rinf)
```

由上述语句得到如下代码：

```
Rinf =
     1     0     3     0
     0     5     0     6
     0     0   Inf     0
     0     0     0    25
```

其中，Rinf'* Rinf 并不等于矩阵 S。

2．一般方阵的高斯消去法

高斯消去法又称 LU 分解，它可以将任意一个方阵 A 分解为一个“心理”下三角矩阵 L 和一个上三角矩阵 U 的乘积，即 $A=LU$。“心理”下三角矩阵定义为下三角矩阵与置换矩阵的乘积。

LU 分解在 MATLAB 7.0 中用函数 lu()来实现，其调用方式如下。

- $[L,U]$ = lu(X)，X 为一个方阵，L 为“心理”下三角矩阵，U 为上三角矩阵，满足关系 $X=L*U$。
- $[L,U,P]$ = lu(X)，X 为一个方阵，L 为下三角矩阵，U 为上三角矩阵，P 为置换矩阵，满足关系 $P*X = L*U$。
- Y = lu(X)，X 为一个方阵。把上三角矩阵和下三角矩阵合并在矩阵 Y 中给出，矩阵 Y 的对角元素为上三角矩阵的对角元素，也即 $Y=L+U-I$。置换矩阵 P 的信息丢失。

考虑线性方程组 $AX=B$，其中，对矩阵 A 可以做 LU 分解，使得 $A=L*U$，这样线性方程组就可以改写成 $L*U*X=B$，由于左除算符'\'可以快速处理三角矩阵，因此可以快速解出：

$$X=U\backslash(L\backslash B)$$

矩阵的行列式的值和矩阵的逆也可以利用 LU 分解来计算，示例如下：

$$\det(A)=\det(L)*\det(U)$$

$$\mathrm{inv}(A)=\mathrm{inv}(U)*\mathrm{inv}(L)$$

例如，做矩阵的 LU 分解，具体代码如下：

```
A=[1 4 2;5 6 9;4 1 8];
[L,U,P]=lu(A)
```

由上述语句得到如下代码：

```
L =
    1.0000         0         0
    0.8000    1.0000         0
    0.2000   -0.7368    1.0000
U =
```

```
    5.0000    6.0000    9.0000
         0   -3.8000    0.8000
         0         0    0.7895
P =
     0     1     0
     0     0     1
     1     0     0
```

对于稀疏矩阵，MATLAB 7.0 提供了函数 luinc()来做不完全 LU 分解，其调用格式如下。

- [*L U*]= luinc(*X*,*DROPTOL*)，其中 ***X***、***L*** 和 ***U*** 的含义与函数 lu()中的变量相同，*DROPTOL* 为不完全 LU 分解的丢失容限。当 *DROPTOL* 设为 0 时，退化为完全 LU 分解。
- [*L U*] = luinc(*X*,*OPTS*)，其中 *OPTS* 为结构体，它有 4 个属性，即 *DROPTOL*、*MICHOL*、*RDIAG* 和 *THRESH*。*DROPTOL* 为不完全 LU 分解的丢失容限；*MICHOL* 为 1 时采用改进算法的不完全 LU 分解，否则不采用改进算法；*RDIAG* 为 1 时，*R* 的对角元素中的零值替换成为 *DROPTOL* 的平方根，为 0 时不做此替换；*THRESH* 是绕对角线旋转因子，其值范围是[0,1]，当 *THRESH* 为 0 时强制绕对角线旋转，*THRESH* 的默认值是 1。
- [*L*,*U*,*P*] = luinc(*X*,'0')，0 级不完全 LU 分解。
- [*L*,*U*] = luinc(*X*,'0')，0 级不完全 LU 分解。
- *Y* = luinc(*X*,'0')，0 级完全 LU 分解。

例如，计算一个稀疏矩阵的不完全分解，并观察其非零元素的个数，具体代码如下：

```
load west0479;               %加载 MATLAB 7.0 自带的一个稀疏矩阵
S = west0479;
[L,U,P] = luinc(S,'0');      %做不完全 LU 分解
figure(1);                   %新开一个图形显示窗口
subplot(2,3,1);              %把该窗口分成 2*3 个字窗口，并在第 1 个字窗口上画图
spy(S);                      %显示稀疏矩阵 S
title('S');                  %给图添上标题
subplot(2,3,2);              %在第 2 个字窗口上画图
spy(L);                      %显示稀疏矩阵 L
title('L');                  %给图添上标题
subplot(2,3,3);              %在第 2 个字窗口上画图
spy(U);                      %显示稀疏矩阵 U
title('U');                  %给图添上标题
subplot(2,3,4);              %在第 2 个字窗口上画图
spy(P*S);                    %显示稀疏矩阵 P*S
title('P*S');                %给图添上标题
subplot(2,3,5);              %在第 2 个字窗口上画图
spy(L*U);                    %显示稀疏矩阵 L*U
title('L*U');                %给图添上标题
```

由上述代码得到如图 3-2 所示的结果。由函数 luinc()得到上三角矩阵 ***U*** 和下三角矩阵 ***L***，

并且 $P*S$ 与 $L*U$ 有相似的稀疏结构。

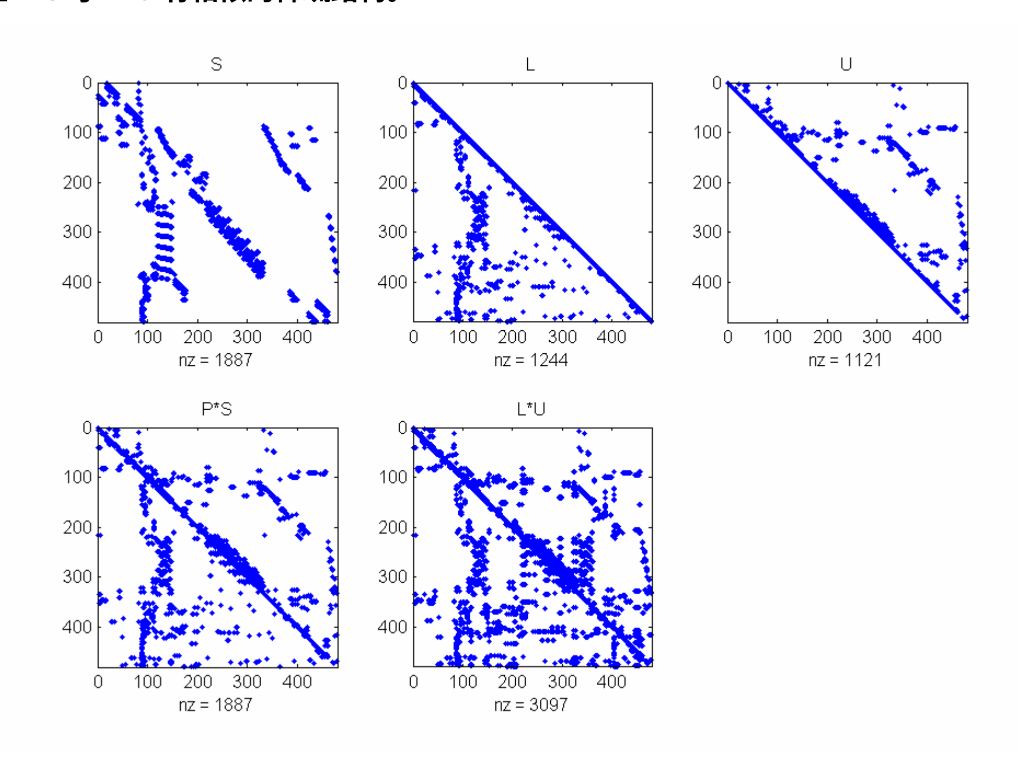


图 3-2　不完全 LU 分解结果

3．矩形矩阵的正交分解

矩形矩阵的正交分解又称 QR 分解。QR 分解把一个 $m \times n$ 的矩阵 A 分解为一个正交矩阵 Q 和一个上三角矩阵 R 的乘积，即 $A=Q*R$。

在 MATLAB 7.0 中 QR 分解由函数 qr()来实现，下面介绍 QR 分解的调用方式。

- [Q,R] = qr(A)，其中矩阵 R 为与矩阵 A 具有相同大小的上三角矩阵，Q 为正交矩阵。它们满足 $A=Q*R$。该调用方式适用于满矩阵和稀疏矩阵。
- [Q,R] = qr(A,0)，为“经济”方式的 QR 分解。设矩阵 A 是一个 $m \times n$ 的矩阵，若 $m>n$，则只计算矩阵 Q 的前 n 列元素，R 为 $n \times n$ 的矩阵；若 $m<=n$，则与[Q,R] = qr(A)效果一致。该调用方式适用于满矩阵和稀疏矩阵。
- [Q,R,E] = qr(A)，R 是上三角矩阵，Q 为正交矩阵，E 为置换矩阵。它们满足 $A*E=Q*R$，程序选择一个合适的矩阵 E 使得 abs(diag(R))是降序排列的。该调用方式适用于满矩阵。
- [Q,R,E] = qr(A,0)，“经济”方式的 QR 分解，其中 E 是一个置换矢量。它们满足 $A(:,E) = Q*R$。该调用方式适用于满矩阵。
- R = qr(A)，返回上三角矩阵 R，这里 R= chol(A'*A)。该调用方式适用于稀疏矩阵。
- R = qr(A,0)，以“经济”方式返回上三角矩阵 R。
- [C,R] = qr(A,B)，其中矩阵 B 必须与矩阵 A 具有相同的行数，矩阵 R 是上三角矩阵，$C=Q'*B$。

例如，通过 QR 分解分析矩阵的秩，具体代码如下：

```
A =   [ 1        2        3
          4        5        6
          7        8        9
         10       11       12];
A_rank=rank(A);
disp(['矩阵 A 的秩 = ' num2str(A_rank)]);
[Q,R]=qr(A)
```

由上述语句得到如下代码：

```
矩阵 A 的秩 = 2
Q =
    -0.0776    -0.8331     0.5473    -0.0221
    -0.3105    -0.4512    -0.7133     0.4373
    -0.5433    -0.0694    -0.2153    -0.8085
    -0.7762     0.3124     0.3813     0.3932
R =
  -12.8841   -14.5916   -16.2992
         0    -1.0413    -2.0826
         0          0    -0.0000
         0          0          0
```

矩阵 ***R*** 的第三行和第四行的元素为 0，可以看出矩阵 ***R*** 不满秩，从而得到矩阵 ***A*** 的秩为 2。该结果与用函数 rank()得到的结果一致。

4．奇异值分解

奇异值分解在矩阵分析中占有极其重要的地位，即对于 $m\times n$ 的矩阵 ***A***，若存在 $m\times m$ 的酉矩阵 ***U*** 和 $n\times n$ 的酉矩阵 ***V***，使得 $A=U*f\dagger*V'$，其中 *f†*为一个 $m\times n$ 的非负对角矩阵，并且其对角元素的值按降序排列，则式子 $A=U*f\dagger*V'$即为矩阵 ***A*** 的奇异值分解，***U***、*f†*和 ***V*** 称为矩阵 ***A*** 的奇异值分解的三对组。

在 MATLAB 7.0 中奇异值分解由函数 svd()来实现，其调用方式如下。

- [*U*,*S*,*V*] = svd(*X*)，奇异值分解。
- [*U*,*S*,*V*] = svd(*X*,0)，“经济”方式的奇异值分解。设矩阵 ***X*** 的大小为 $m\times n$，若 $m>n$，则只计算 ***U*** 的前 n 列元素，***S*** 为 $n\times n$ 的矩阵；若 $m<=n$，则与[*U*,*S*,*V*] = svd(*X*)效果一致。
- [*U*,*S*,*V*] = svd(*X*,'econ')，设矩阵 ***X*** 的大小为 $m\times n$，若 $m>=n$，则与[*U*,*S*,*V*] = svd(*X*,0)效果一致，若 $m<n$，则只计算 ***V*** 的前 m 列元素，***S*** 为 $m\times m$ 的矩阵。
- *s* = svd(*X*)，返回包含所有奇异值的矢量。

例如，把矩阵 ***A*** 分解为奇异值的三对组，具体代码如下：

```
A=[1 2;3 4;5 6];
[U S V]=svd(A)
```

由上述语句得到如下代码：

```
U =
```

```
    -0.2298     0.8835     0.4082
    -0.5247     0.2408    -0.8165
    -0.8196    -0.4019     0.4082
S =
     9.5255          0
          0     0.5143
          0          0
V =
    -0.6196    -0.7849
    -0.7849     0.6196
```

如果采用“经济”方式来求奇异值，具体代码如下：

```
A=[1 2;3 4;5 6];
[U S V]=svd(A,0)
```

由上述语句得到如下代码：

```
U =
    -0.2298     0.8835
    -0.5247     0.2408
    -0.8196    -0.4019
S =
     9.5255          0
          0     0.5143
V =
    -0.6196    -0.7849
    -0.7849     0.6196
```

由上面的语句看出用“经济”方式得到的矩阵尺寸更小。

MATLAB 7.0 还支持做一般奇异值分解运算。MATLAB 7.0 中用函数 gsvd()来进行一般奇异值分解，其基本调用格式如下：

- [*U*,*V*,*X*,*C*,*S*] = gsvd(*A*,*B*)，其中矩阵 *A* 和矩阵 *B* 必须拥有相同的列数，*U* 和 *V* 是酉矩阵，*X* 一般是一个方阵，*C* 和 *S* 是非负对角矩阵。它们之间的关系如下：*A* = *U***C***X*’，*B* = *V***S***X*’，*C***C* + *S***S* = *I*。若 *A* 和 *B* 分别是 $m\times p$ 和 $n\times p$ 的矩阵，则 *U* 是 $m\times m$ 的矩阵，*V* 是 $n\times n$ 的矩阵，*X* 是 $q\times q$ 的方阵，其中 $q=\min(m+n,p)$。
- sigma = gsvd(*A*,*B*)，返回包含所有一般奇异值的矢量，它等于 qrt(diag(*C***C*)./diag(*S***S*))。
- gsvd(*A*,*B*,0)，“经济”方式分解一般奇异值。

5．舒尔分解

舒尔分解定义式为：

$$A=U*S*U'$$

其中 *A* 必须是一个方阵，*U* 是一个酉矩阵，*S* 是一个块对角化矩阵，由对角线上的 1×1 和 2×2 块组成。特征值可以由矩阵 *S* 的对角块给出，而矩阵 *U* 给出比特征向量更多的数值特

征。此外，对缺陷矩阵也可以进行舒尔分解。

MATLAB 7.0 中用函数 schur()来进行舒尔分解，其调用格式如下。

- [*U*,*S*] = schur(*A*)，返回酉矩阵 ***U*** 和块对角矩阵 ***S***。
- *S* = schur(*A*)，仅返回块对角矩阵 ***S***。

如果矩阵 ***A*** 是一个实矩阵，则有两种求解方式。

- schur(*A*,'real')，返回的实特征值放在对角线上，而把复特征值放在对角线上的 2×2 块中。
- schur(*A*,' complex')，返回的矩阵 ***S*** 是上三角矩阵，并且如果矩阵 ***A*** 有复特征值，则矩阵 ***S*** 是复矩阵。

函数 rsf2csf()可以把实数形式的舒尔矩阵转换成复数形式的舒尔矩阵。

例如，对 4 阶魔方矩阵做舒尔矩阵，具体代码如下：

```
A=magic(4);% 4 阶魔方矩阵
[U S]=schur(A)
```

由上述语句得到如下代码：

```
U =
    -0.5000    -0.8236    -0.1472    -0.2236
    -0.5000     0.4236     0.3472    -0.6708
    -0.5000     0.0236     0.5472     0.6708
    -0.5000     0.3764    -0.7472     0.2236
S =
    34.0000     0.0000     0.0000     0.0000
          0     8.9443    13.4164    -0.0000
          0          0    -8.9443     0.0000
          0          0          0    -0.0000
```

3.1.4 矩阵的特征值和特征向量

一个 $n×n$ 的方阵 ***A*** 的特征值 λ 和特征向量 v 满足下列关系式的变量：

$$A*v=\lambda*v$$

其中，λ 为一个标量，***v*** 为一个向量。如果把矩阵 ***A*** 的所有 n 个特征值放在矩阵 ***D*** 的对角线上，相应的特征向量按照与特征值对应的顺序排列，作为矩阵 ***V*** 的列，特征值问题可以改写为：

$$A*V=V*D$$

如果 ***V*** 是非奇异的，该问题可以认为是一个特征值分解问题，此时关系式如下：

$$A= V*D*V^{-1}$$

广义特征值问题是指方程 $A*X=\lambda*B*X$ 的非平凡解问题。其中 ***A*** 和 ***B*** 都是 $n×n$ 的矩阵，λ 是一个标量。满足方程的 λ 称为广义特征值，对应的向量 X 称为广义特征向量。

MATLAB 7.0 中用函数 eig()来进行求特征值和特征向量，其调用格式如下。

- d = eig(*A*)，返回矩阵 ***A*** 的所有特征值。

- [*V*,*D*] = eig(*A*)，返回矩阵 ***A*** 的特征值和特征向量，它们满足如下关系：*A**V* = *V**D*
- [*V*,*D*] = eig(*A*,'nobalance')，在求解特征值和特征向量时不采用初期的平衡步骤。一般来说平衡步骤对输入矩阵进行调整，这使得计算出的特征值和特征向量更加准确。然而如果输入矩阵中确实含有值很小的元素（可能会导致截断误差），平衡步骤有可能加大这种误差，从而得到错误的特征值和特征向量。
- *d* = eig(*A*,*B*)，返回矩阵 ***A*** 和 ***B*** 的广义特征值。
- [*V*,*D*] = eig(*A*,*B*)，返回矩阵 ***A*** 和 ***B*** 的广义特征值和广义特征向量。
- [*V*,*D*] = eig(*A*,*B*,*flag*)，*flag* 有'chol'和'qz'两种值。当 *flag*='chol'时，计算广义特征值采用 ***B*** 的 Cholesky 分解来实现。当 *flag*='qz'时，无论矩阵的对称性如何，都采用 QZ 算法来求解广义特征值。

例如，求解矩阵 ***A*** 的特征值和特征向量，具体代码如下：

```
A = [ 6       12       19
        -9     -20     -33
          4        9       15 ];
[V D]=eig(A)
```

由上述语句得到如下代码：

```
V =
   -0.4741                    -0.4082 - 0.0000i    -0.4082 + 0.0000i
     0.8127                      0.8165                      0.8165
   -0.3386                    -0.4082 + 0.0000i    -0.4082 - 0.0000i
D =
   -1.0000                              0                              0
              0                   1.0000 + 0.0000i              0
              0                              0                   1.0000 - 0.0000i
```

由上式可以看出矩阵 ***A*** 的特征值中有两个是相同的，与之对应的是矩阵 ***A*** 的特征向量也有两个是相同的。故矩阵 ***V*** 是奇异矩阵，该矩阵不可以做特征值分解。

对于稀疏矩阵，MATLAB 7.0 提供函数 eigs()来计算模最大的前 *k* 个特征值，该函数的具体用法参见 METLAB7.0 的在线帮助。

3.1.5　非线性矩阵运算

MATLAB 7.0 提供的非线性矩阵运算函数及其功能如表 3-4 所示。

表 3-4　　三角函数

函数名	功能描述
expm	矩阵指数运算
logm	矩阵对数运算
sqrtm	矩阵开平方运算
funm	一般非线性矩阵运算

下面将详细介绍这几个函数的用法。

1．矩阵指数运算

以一个线性微分方程组为例进行讲解，示例如下：

$$\frac{dx(t)}{dt} = Ax(t)$$

其中 $x(t)$是与时间有关的一个向量，A 是与时间无关的矩阵。该方程的解可以表示为：

$$x(t) = e^{tA}x(0)$$

因此，解一个线性微分方程组的问题就等效于计算矩阵指数运算。在 MATLAB 7.0 中函数 expm()用于计算矩阵指数运算。其调用格式如下。

Y = expm(X)，返回矩阵 X 的指数

例如，一个三元的线性方程组，其矩阵 A 为 3×3 的矩阵，初值 $x(0)$为 3×1 的向量，求该线性方程组的解，具体代码如下：

```
A = [0      -6      -1;
       6       2     -16;
      -5      20     -10];                          %矩阵 A
x0=[1;1;1];                                          %初值 x(0)
t=0:0.01:1;                                          %时间 t
xt=[];                                               %计算结果 x(t)
for i=1:length(t),                                   %计算每个时间点上的 x(t)
      xt(i,:)=expm(t(i)*A)*x0;
end;
plot3(xt(:,1),xt(:,2),xt(:,3),'-o')                  %画三维图显示 x(t)
grid on;                                             %给图添加辅助网格
xlabel('xt(:,1)');                                   %给图的坐标轴 x 命名
ylabel('xt(:,2)');                                   %给图的坐标轴 y 命名
zlabel('xt(:,3)');                                   %给图的坐标轴 z 命名
title('矩阵指数函数结果');                            %给图添加标题
```

由上述语句得到如图 3-3 所示的结果图。

> 注意：矩阵指数运算函数 expm()是对整个矩阵作指数运算，它有别于矩阵元素的指数运算 exp()，具体内容将在 3.2.2 节中介绍。

2．矩阵对数运算

矩阵对数运算是矩阵指数运算的逆运算，在 MATLAB 7.0 中函数 logm()用来实现矩阵对数运算，其调用格式如下。

- L = logm(A)，返回矩阵 A 的对数 L。
- [L, *exitflag*] = logm(A)，返回矩阵 A 的对数 L，同时返回标量 *exitflag*。当 *exitflag* 为 0 时，函数成功运行；当 *exitflag* 为 1 时，表明运算过程中某些泰勒级数不收敛，但运算结果仍可能是精确的。

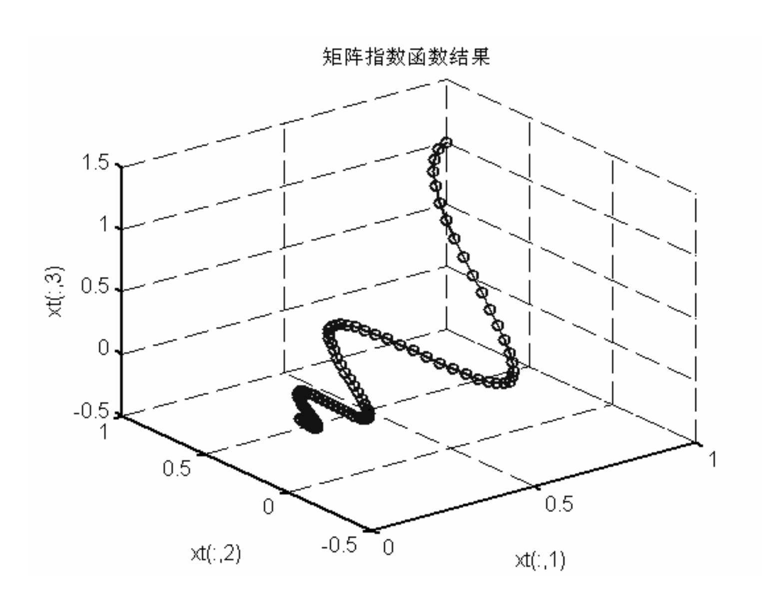


图 3-3　矩阵指数函数结果

例如，下面例子说明函数 logm()是函数 expm()的反函数，具体代码如下：

```
A=[1 2 3;4 5 6;7 8 9];
Y=expm(A)
A_expm_logm=logm(Y)
```

由上面的语句得到如下代码：

```
Y =
   1.0e+006 *
      1.1189      1.3748      1.6307
      2.5339      3.1134      3.6929
      3.9489      4.8520      5.7552
A_expm_logm =
      1.0000      2.0000      3.0000
      4.0000      5.0000      6.0000
      7.0000      8.0000      9.0000
```

由结果可以看出函数 logm()是函数 expm()的反函数。

> 注意：矩阵对数运算函数 logm()是对整个矩阵作对数运算，它有别于将在 3.2.2 小节中介绍的矩阵元素的对数运算 log()。

3．矩阵开平方运算

对矩阵 A 开平方得到矩阵 X，满足 $X*X=A$。如果矩阵 A 的某个特征值具有负实部，则其平方根 X 为复数矩阵。如果矩阵 A 是奇异的，则它有可能不存在平方根 X。在 MATLAB 7.0 中有两种计算矩阵平方根的方法：A^0.5 和 sqrtm(A)。函数 sqrtm()比 A^0.5 的运算精度更高。函数 sqrtm()的调用方法如下：

- X = sqrtm(A)，返回矩阵 A 的平方根 X，如果矩阵 A 是奇异的，将返回警告信息。
- [X,*resnorm*] = sqrtm(A)，不返回任何警告信息，返回留数 norm(A-X^2,'fro')/norm(A,'fro')
- [X,*alpha*,*condest*] = sqrtm(A)，返回一个稳定因子 *alpha*，以及矩阵 X 的条件数估计 *condest*。

例如，求一个正定矩阵 A 的平方根 X，并验证 $X*X=A$，具体代码如下：

```
A=[    5      -4        1;
       -4        6      -4;
        1      -4        6];                       %矩阵 A
X=sqrtm(A)                                          %矩阵的平方根 X
XX=X*X                                                %X*X
```

由上述语句得到如下代码：

```
X =
       2.0091     -0.9812       0.0223
     -0.9812       2.0463     -0.9218
       0.0223     -0.9218       2.2693
XX =
       5.0000     -4.0000       1.0000
     -4.0000       6.0000     -4.0000
       1.0000     -4.0000       6.0000
```

由 $X*X$ 与矩阵 A 相等，验证了运算的正确性。

> 注意：矩阵开平方运算函数 sqrtm()是对整个矩阵作开平方运算，它有别于对矩阵元素的开平方运算.^0.5。

4．一般非线性矩阵运算

MATLAB 7.0 提供了计算一般非线性矩阵运算的函数 funm()，其基本调用格式如下。

- F = funm(A,fun)，把函数 fun 作用在方阵 A 上，其中 fun 是一个函数句柄。函数 fun 的格式是 fun(X,k)，其中 X 是一个向量，k 是一个标量。fun(X,k)返回的值应该是 fun 代表的函数的 k 阶导数作用在向量 X 的值。fun 代表的函数的泰勒展开在无穷远处必须是收敛的（函数 logm()是惟一的例外）。

MATLAB 7.0 中可以使用的函数 fun 如表 3-5 所示。

表 3-5　　一般非线性矩阵运算函数

函数名	调用格式
exp	funm(A, @exp)
log	funm(A, @log)
sin	funm(A, @sin)
cos	funm(A, @cos)
sinh	funm(A, @sinh)
cosh	funm(A, @cosh)

计算矩阵的指数时，表达式 funm(A, @exp)和 expm(A)中哪种方式的计算精度更高取决于矩阵 A 本身的性质。

例如，计算矩阵 *A* 的正弦函数，具体代码如下：

```
A=[   5     -4       1;
     -4      6     -4;
      1     -4      6];                    %矩阵 A
A_sin=funm(A,@sin)                         %计算矩阵 A 的正弦函数
```

由上述语句得到计算如下代码：

```
A_sin =
   -0.4355    0.4693    0.4544
    0.4693   -0.0192    0.3168
    0.4544    0.3168   -0.5909
```

3.2　矩阵元素的数学函数

本小节介绍矩阵元素的数学函数，包括三角函数、指数对数函数、复数函数、截断函数和求余函数。这些函数共同的特点是函数的运算都是针对矩阵的元素，即它们都是对矩阵中的每个元素都进行运算。

3.2.1　三角函数

MATLAB 7.0 提供的三角函数及其功能如表 3-6 所示。

表 3-6　三角函数

函数名	功能描述
sin	正弦
sind	正弦，输入以度为单位
sinh	双曲正弦
asin	反正弦
asind	反正弦，输出以度为单位
asinh	反双曲正弦
cos	余弦
cosd	余弦，输入以度为单位
cosh	双曲余弦
acos	反余弦
acosd	反余弦，输出以度为单位
acosh	反双曲余弦
tan	正切
tand	正切，输入以度为单位
tanh	双曲正切
atan	反正切
atand	反正切，输出以度为单位
atan2	四象限反正切
atanh	反双曲正切
sec	正割

续表

函数名	功能描述
secd	正割，输入以度为单位
sech	双曲正割
asec	反正割
asecd	反正割，输出以度为单位
asech	反双曲正割
csc	余割
cscd	余割，输入以度为单位
csch	双曲余割
acsc	反余割
acscd	反余割，输出以度为单位
acsch	反双曲余割
cot	余切
cotd	余切，输入以度为单位
coth	双曲余切
acot	反余切
acotd	反余切，输出以度为单位
acoth	反双曲余切

例如，计算 0°~360°的正弦函数、余弦函数和它们平方和的值，具体代码如下：

```
x=0:10:360;                                          %角度 0°~360°
figure(1);                                           %新开一个画图窗口
square_sum=sind(x).^2+cosd(x).^2;                    %正弦函数和余弦函数的平方和
plot(x,sind(x),'ro-',x,cosd(x),'g+-',x,square_sum,'bd-');   %画图
xlabel('角度');                                      %标注横坐标
ylabel('函数值');                                    %标注纵坐标
legend('正弦函数','余弦函数','sin(x)^2+cos(x)^2');   %给图添加图例
```

由上述语句得到如图 3-4 所示的结果图。

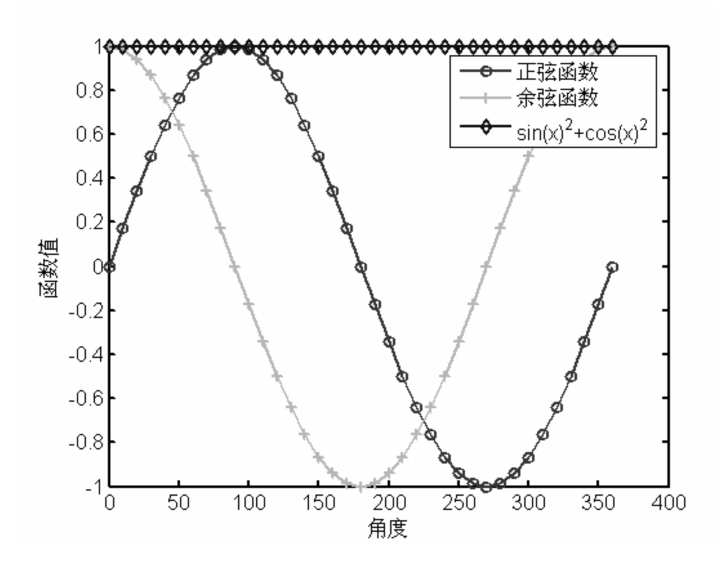


图 3-4 三角函数示意图

3.2.2　指数和对数函数

MATLAB 7.0 提供的指数、对数函数及其功能如表 3-7 所示。

表 3-7　　指数和对数函数

函数名	功能描述	函数名	功能描述
exp	指数	realpow	对数，若结果是复数则报错
expm1	准确计算 exp(x)-1 的值	reallog	自然对数，若输入不是正数则报错
log	自然对数（以 e 为底）	realsqrt	开平方根，若输入不是正数则报错
log1p	准确计算 log(1+x) 的值	sqrt	开平方根
log10	常用对数（以 10 为底）	nthroot	求 x 的 n 次方根
log2	以 2 为底的对数	nextpow2	返回满足 2^P >= abs(N)的最小正整数 P，其中 N 为输入
pow2	以 2 为底的指数		

例如，计算 e^x 和 2^x 的值，具体代码如下：

```
x=-1:0.1:4;                          %x 值
figure(1);                           %新开一个画图窗口
plot(x,exp(x),'ro-',x,pow2(x),'g+-'); %画图
xlabel('x');                         %标注横坐标
ylabel('函数值');                     %标注纵坐标
legend('e^x','2^x');                 %给图添加图例
```

由上述语句得到如图 3-5 所示的示意图。

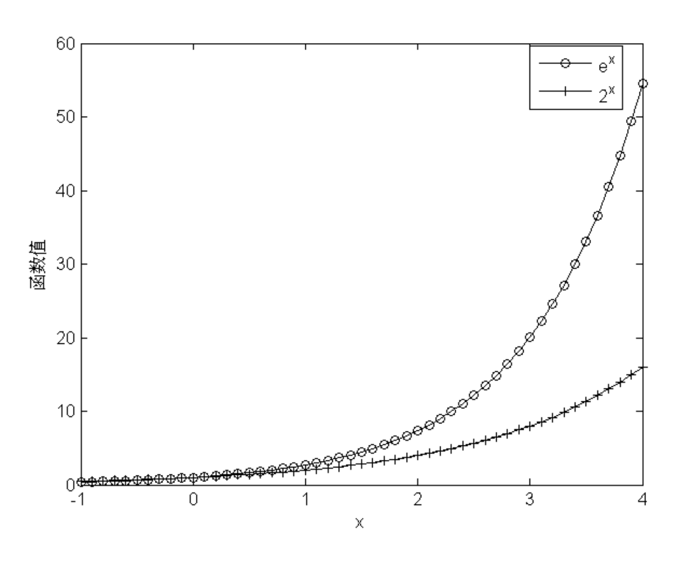


图 3-5　指数函数示意图

3.2.3　复数函数

MATLAB 7.0 提供的复数函数及其功能如表 3-8 所示。

表 3-8　复数函数

函数名	功能描述	函数名	功能描述
abs	绝对值（复数的模）	real	复数的实部
angle	复数的相角	unwrap	调整矩阵元素的相位
complex	用实部和虚部构造一个复数	isreal	是否为实数矩阵
conj	复数的共轭	cplxpair	把复数矩阵排列成为复共轭对
imag	复数的虚部		

复数函数中除了函数 unwrap()和 cplxpair()的用法比较复杂外，其他函数都比较简单。下面就详细介绍函数 unwrap()和 cplxpair()。

函数 unwrap()用于对表示相位的矩阵进行校正，当矩阵相邻元素的相位差大于设定阈值（默认值为 π）时，函数 unwrap()通过加±2 π来校正相位。函数 unwrap()的基本调用格式如下。

- Q = unwrap(P)，当相位大于默认阈值 π时，矫正相位；
- Q = unwrap(P,*tol*)，用 *tol* 来设定阈值；
- Q = unwrap(P,[],*dim*)，用默认阈值 π在给定维 *dim* 上做相位矫正；
- Q = unwrap(P,*tol*,*dim*)，用阈值 *tol* 在给定维 *dim* 上做相位矫正。

例如，下面数据是滤波器的相位响应，其中 *w* 为频率，*p* 为相位。下面程序将比较数据校正前和程序后的差别，具体代码如下：

```
w = 1000*[0.2:.2:3,3.5:1:10];                                    %频率
p = [-1.5728 -1.5747 -1.5772 -1.5790 -1.5816 -1.5852 -1.5877 -1.5922 ...
     -1.5976 -1.6044 -1.6129 -1.6269 -1.6512 -1.6998 -1.8621 1.7252 ...
      1.6124    1.5930    1.5916    1.5708    1.5708    1.5708 ]; %相位
  figure(1);                                                     %开一个新的画图窗口
  semilogx(w,p,'ro-');                                           %用半对数坐标画原始数据
  hold on;                                                       %保持原来的图
  semilogx(w,unwrap(p),'b*-');                                   %用半对数坐标画校正后数据
  xlabel('频率');                                                %添加横坐标说明
  ylabel('相位');                                                %添加纵坐标说明
  legend('原始相频特性曲线','矫正后的相频特性曲线');              %添加图例
```

程序得到结果如图 3-6 所示。由图可见，原始数据中频率 3000Hz~3500Hz 之间，相位变化了 3.5873（单位：弧度），大于默认的阈值，因此 unwrap()函数通过把相位减去 2 π从而把相位跳变校正为 2.6959。

函数 cplxpair()把复数数组排列成复数的共轭对。函数 cplxpair()的基本调用格式如下。

- B = cplxpair(A)，以默认的误差容限 100*eps 寻找复数的共轭对；
- B = cplxpair(A,*tol*)，以指定的误差容限 *tol* 寻找复数的共轭对；
- B = cplxpair(A,[],*dim*)，在 *dim* 维度上，以默认的误差容限寻找复数的共轭对；
- B = cplxpair(A,*tol*,*dim*)，在 *dim* 维度上，以指定的误差容限 *tol* 寻找复数的共轭对。

如果输入矩阵的复数个数是奇数或者在误差容限内复数不能组合成共轭对，MATLAB 7.0 将返回如下错误信息：

```
Complex numbers can't be paired.
```

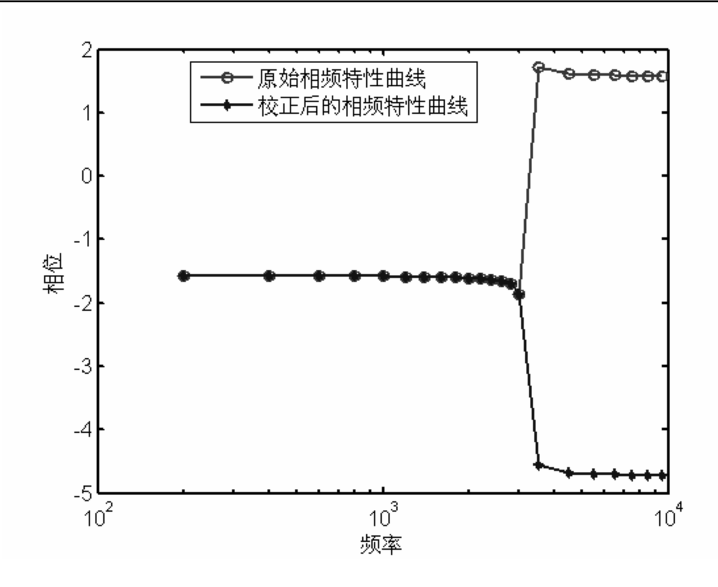


图 3-6　unwrap()函数

例如，把复数数组排列成复数的共轭对，具体代码如下：

```
A=[1+i,2,0.5-3i,5+4i,6,1-i,5-4i,0.5+3i];                %复数矩阵
cplxpair(A)                                              %排列成共轭对
```

由上述语句输出如下代码：

```
ans =
  Columns 1 through 4
    0.5000 - 3.0000i    0.5000 + 3.0000i    1.0000 - 1.0000i    1.0000 + 1.0000i
  Columns 5 through 8
    5.0000 - 4.0000i    5.0000 + 4.0000i    2.0000              6.0000
```

3.2.4　截断和求余函数

MATLAB 7.0 提供的截断和求余函数及其功能如表 3-9 所示。

表 3-9　截断和求余函数

函数名	功能描述	函数名	功能描述
fix	向零取整	mod	除法求余（与除数同号）
floor	向负无穷方向取整	rem	除法求余（与被除数同号）
ceil	向正无穷方向取整	sign	符号函数
round	四舍五入		

采用下面的例子来说明函数 fix()、floor()、ceil()和 round()的区别，具体代码如下：

```
a=[-1.55 -1.45 1.45 1.55];
a_fix=fix(a)
a_floor=floor(a)
a_ceil=ceil(a)
a_round=round(a)
```

由上述语句得到如下代码：

```
a_fix =
     -1    -1     1     1
a_floor =
     -2    -2     1     1
a_ceil =
     -1    -1     2     2
a_round =
     -2    -1     1     2
```

用下面的例子说明函数 mod()和 rem()的区别，具体代码如下：

```
a=[10 -10 10 -10];
b=[8 -8 -8 8];
c_rem=rem(a,b)
c_mod=mod(a,b)
```

由上述语句得到如下代码：

```
c_rem =
      2    -2     2    -2
c_mod =
      2    -2    -6     6
```

求矩阵的符号使用函数 sign()，示例代码如下：

```
a=-2:2;
a_sign=sign(a)
```

由上述语句得到如下代码：

```
a_sign =
     -1    -1     0     1     1
```

3.3 特殊数学函数

本小节介绍一些用途比较特殊的数学函数，包括特殊函数、数论函数和坐标变换函数。

3.3.1 特殊函数

特殊函数通常是数学物理方程的解，并且经常在数学、物理和工程问题中出现。MATLAB 7.0 提供的特殊函数及其功能如表 3-10 所示。

表 3-10　特殊函数

函数名	功能描述	函数名	功能描述
airy	Airy 函数	erfc	余误差函数：erfc(x)=1-erf(x)
besselj	第一类 Bessel 函数	erfcx	erfcx(x) = exp(x^2) * erfc(x)
bessely	第二类 Bessel 函数	erfinv	误差函数的逆函数

续表

函数名	功能描述	函数名	功能描述
besselh	第三类 Bessel 函数	expint	指数积分函数
besseli	第一类 Bessel 函数	gamma	Gamma 函数
besselk	第二类改进的 Bessel 函数	gammainc	不完全 Gamma 函数
beta	Beta 函数	gammaln	对数 Gamma 函数
betainc	不完全 Beta 函数	psi	多 Γ（Polygamma）函数
betaln	对数 Beta 函数	legendre	连带勒让德函数
ellipj	Jacobi 椭圆函数	cross	矢量叉乘
ellipke	完全椭圆积分	dot	矢量点乘
erf	误差函数		

在后面的章节将详细介绍这些特殊函数的定义和用法。为了阅读方便，把这些特殊函数进行分类，包括：Airy 函数、Bessel 函数、Gamma 函数、Beta 函数、Jacobi 椭圆函数和完全椭圆积分、误差函数、指数积分函数和连带勒让德函数。

1．Airy 函数

Airy 函数是微分方程 $\dfrac{d^2W}{dZ^2}-ZW=0$ 的解。有两类 Airy 函数：第一类 Airy 函数 Ai(Z) 和第二类 Airy 函数 Bi(Z)。它们可以用改进的第一类 Bessel 函数 $I_v(Z)$和改进的第二类 Bessel 函数 $K_v(Z)$来定义，表达式如下：

$$\mathrm{Ai}(Z)=\left[\frac{1}{\pi}\sqrt{\frac{Z}{3}}\right]\mathrm{I}_{\frac{1}{3}}(\frac{2}{3}Z^{\frac{3}{2}}),$$

$$\mathrm{Bi}(Z)=\sqrt{\frac{Z}{3}}\left[\mathrm{K}_{-\frac{1}{3}}(\frac{2}{3}Z^{\frac{3}{2}})+\mathrm{K}_{\frac{1}{3}}(\frac{2}{3}Z^{\frac{3}{2}})\right]$$

函数 Airy ()的调用方式如下：

- W = Airy(Z)，返回第一类 Airy 函数 Ai(Z)；
- W = Airy(0,Z)，与 Airy (Z)相同；
- W = Airy(1,Z)，返回第一类 Airy 函数 Ai(Z)的导数 Ai’(Z)；
- W = Airy (2,Z)，返回第二类 Airy 函数 Bi(Z)；
- W = Airy (3,Z)，返回第二类 Airy 函数 Bi(Z)的导数 Bi’(Z)。

2．Bessel 函数

Bessel 函数是微分方程 $Z^2\dfrac{d^2y}{dz^2}+Z\dfrac{dy}{dZ}+(Z^2-v^2)y=0$ 的解，其中 v 是常量。该方程有两个线性无关的解，第一类 Bessel 函数 $\mathrm{J}_v(Z)$ 和第二类 Bessel 函数 $\mathrm{Y}_v(Z)$，它们的表达式如下：

$$J_v(Z)=\left(\frac{Z}{2}\right)^v\sum_{k=0}^{\infty}\frac{(Z)^k}{k!\Gamma(v+k+1)}$$

$$Y_v(Z)=\frac{J_v(Z)\cos(v\pi)-J_{-v}(Z)}{\sin(v\pi)}$$

有时也采用 Hankel **函数来表示** Bessel **方程，它们是第一类** Bessel **函数和第二类** Bessel **函数的线性组合：**

$H_v^{(1)}(Z)=J_v(Z)+iY_v(Z)$，

$H_v^{(1)}(Z)=J_v(Z)-iY_v(Z)$。

Hankel **函数** $H_v^{(k)}$ **也称为第三类** Bessel **函数。**

在 MATLAB 7.0 **中与** Bessel **函数相关的函数的调用方式如下。**

- *J* = besselj(*nu*,*Z*)，**返回第一类** Bessel **函数** $J_v(Z)$；
- *J* = besselj(*nu*,*Z*,1)，besselj(*nu*,*Z*).*exp(-abs(imag(*Z*)))；
- *Y* = bessely(*nu*,*Z*)，**返回第二类** Bessel **函数** $Y_v(Z)$；
- *Y* = bessely(*nu*,*Z*,1)，**返回** bessely(*nu*,*Z*).*exp(-abs(imag(*Z*)))；
- *H* = besselh(*nu*,*K*,*Z*)，**返回第三类** Bessel **函数** $H_v^{(k)}$；
- *H* = besselh(*nu*,*Z*)，**返回** $H_v^{(1)}$。

改进的 Bessel **函数是微分方程** $Z^2\frac{d^2y}{dZ^2}+Z\frac{dy}{dZ}-(Z^2+v^2)y=0$ **的解，其中** v **是常量。**

该方程有两个线性无关的解：第一类改进的 Bessel **函数** $I_v(Z)$ **和第二类改进的** Bessel **函数** $K_v(Z)$，**它们的表达式如下：**

$$I_v(Z)=\left(\frac{Z}{2}\right)^v\sum_{k=0}^{\infty}\frac{(\frac{Z^2}{4})^k}{k!\Gamma(v+k+1)}$$

$$K_v(Z)=\frac{\pi}{2}\frac{I_{-v}(Z)-I_v(Z)}{\sin(v\pi)}$$

在 MATLAB 7.0 **中与改进的** Bessel **函数相关的函数的调用方式如下。**

- I = besseli(nu,Z)，返回第一类改 2 进的 Bessel 函数 $\mathrm{I}_v(Z)$；
- I = besseli(nu,Z,1)，返回 besseli(nu,Z).*exp(-abs(real(Z)))；
- K = besselk(nu,Z)，返回第二类改进的 Bessel 函数 $\mathrm{K}_v(Z)$；
- K = besselk(nu,Z,1)，返回 besselk(nu,Z).*exp(Z)。

3．Gamma 函数和 Beta 函数

在 MATLAB 7.0 中，Gamma 函数和 Beta 函数的定义如下。

- Gamma 函数：$\Gamma(a)=\int_0^{\infty}\mathrm{e}^{-t}t^{a-1}\mathrm{d}t$；
- 不完全 Gamma 函数：$P(x,a)=\dfrac{1}{\Gamma(a)}\int_0^{x}\mathrm{e}^{-t}t^{a-1}\mathrm{d}t$；
- 多 Γ 函数：$\psi_n(x)=\dfrac{d^{n-1}\psi(x)}{dx^{n-1}}$，其中$\psi(x)=\dfrac{\Gamma'(x)}{\Gamma(x)}=\dfrac{d\ln(\Gamma(x))}{dx}$，$\psi_n(x)$称为$(n+1)\Gamma$ 函数，例如$\psi_3(x)$称为 4Γ 函数；
- Beta 函数：$B(z,w)=\int_0^1 t^{z-1}(1-t)^{w-1}\mathrm{d}t=\dfrac{\Gamma(z)\Gamma(w)}{\Gamma(z+w)}$；
- 不完全 Beta 函数：$I_x(z,w)=\dfrac{1}{B(z,w)}\int_0^x t^{z-1}(1-t)^{w-1}\mathrm{d}t$。

在 MATLAB 7.0 中与 Gamma 函数和 Beta 函数相关的函数的调用方式如下。

- Y = gamma(a)，返回 Gamma 函数 $\Gamma(a)$；
- Y = gammainc(x,a)，返回不完全 Gamma 函数 $\mathrm{P}(x,a)$；
- Y = gammainc(X,A,$tail$)，当 tail='lower'时返回 $\mathrm{P}(x,a)$，当 $tail$='upper'时返回 1-$\mathrm{P}(x,a)$；
- Y = gammaln(A)，返回 $\ln(\Gamma(a))$，可以避免采用 log(gamma(a))造成的溢出情况；
- Y = psi(X)，返回双 Γ 函数 $\psi_1(x)$；
- Y = psi(k,X)，返回 k+2Γ 函数 $\psi_{\mathrm{k}+1}(x)$；

- B = beta(z,w)，返回 Beta 函数 $\mathrm{B}(z,w)$ ；
- I = betainc(x,z,w)，返回不完全 Beta 函数 $\mathrm{I}_x(z,w)$ ；
- L = betaln(z,w)，返回 $\ln(B(z,w))$ ，可以避免采用 log(beta(a))造成的溢出情况。

4．Jacobi 椭圆函数和完全椭圆积分

Jacobi 椭圆函数 sn(u)、cn(u)和 dn(u)是定义在勒让德第一类椭圆积分基础上的。其中，勒让德第一类椭圆积分表达式如下：

$$\mathrm{u}(m,\phi)=\int_0^{\phi}\frac{\mathrm{d}\theta}{\left(1-m\sin^2(\theta)\right)^{\frac{1}{2}}}$$

椭圆积分的反函数定义为 $\phi = \mathrm{am}(u)$ ，则 sn(u)、cn(u)和 dn(u)的表达式为：

$$\mathrm{sn}(u)=\sin(\phi)$$

$$\mathrm{cn}(u)=\cos(\phi)$$

$$\mathrm{dn}(u)=(1-m\sin^2\phi)^{\frac{1}{2}}$$

第一类完全椭圆积分 K(m)也是定义在勒让德第一类椭圆积分基础上的，其表达式如下：

$$\mathrm{K}(m)=\mathrm{u}(m,\frac{\pi}{2})=\int_0^{\frac{\pi}{2}}\frac{\mathrm{d}\theta}{\left(1-m\sin^2(\theta)\right)^{\frac{1}{2}}}$$

第二类完全椭圆积分 E(m)的定义如下：

$$\mathrm{E}(m)=\int_0^{\frac{\pi}{2}}\left(1-m\sin^2(\theta)\right)^{\frac{1}{2}}\mathrm{d}\theta$$

在 MATLAB 7.0 中与 Jacobi 椭圆函数和完全椭圆积分相关的函数的调用方式如下。

- [SN,CN,DN] = ellipj(U,M)，返回 Jacobi 椭圆函数 sn(u)，cn(u)和 dn(u)；
- [SN,CN,DN] = ellipj(U,M,tol)，以指定精度 tol 计算 Jacobi 椭圆函数，默认精度是 eps，若指定一个更大的值会降低计算精度，提高计算速度；
- K = ellipke(M)，返回第一类完全椭圆积分 K(m)；
- [K,E] = ellipke(M)，返回第一类完全椭圆积分 K(m)和第二类完全椭圆积分 E(m)；
- [K,E] = ellipke(M,tol)，以精度指定 tol 计算完全椭圆积分，若指定一个更大的值会降低计算精度，提高计算速度。

5．误差函数

与误差函数有关的表达式如下：

- **误差函数：** $\mathrm{erf}(x)=\frac{2}{\sqrt{\pi}}\int_0^x \mathrm{e}^{-t^2}\mathrm{d}t$
- **余误差函数：** $\mathrm{erfc(x)}=\frac{2}{\sqrt{\pi}}\int_x^\infty \mathrm{e}^{-t^2}\mathrm{dt}=1-\mathrm{erf}(x)$

在 MATLAB 7.0 **中与误差函数相关的函数的调用方式如下。**

- *Y* = erf(*X*)，**返回误差函数；**
- *Y* = erfc(*X*)，**返回余误差函数；**
- *Y* = erfcx(*X*)，**返回** $\mathrm{e}^{x^2}\mathrm{erfc(x)}$ **；**
- *X* = erfinv(*Y*)，**返回误差函数的反函数；**
- *X* = erfcinv(*Y*)，**返回余误差函数的反函数。**

6．**指数积分函数**

指数积分表达式为： $E_1(x)=\int_x^\infty \frac{\mathrm{e}^{-t}}{t}\mathrm{d}t$

在 MATLAB 7.0 **中用函数** expint()**计算指数积分，其调用格式如下。**
Y = expint(*X*)，**返回指数积分** $E_1(x)$。

7．**连带勒让德函数**

连带勒让德函数是连带勒让德方程 $(1-x^2)\frac{\mathrm{d}^2y}{\mathrm{d}x^2}-2x\frac{\mathrm{d}y}{\mathrm{d}x}+\left(n(n+1)-\frac{m^2}{1-x^2}\right)y=0$ **的解，记为** $\mathrm{P}_n^m(x)$ **。连带勒让德函数可以用勒让德多项式来表示，表达式如下：**

$$\mathrm{P}_n^m(x)=(-1)^m(1-x^2)^{\frac{m}{2}}\frac{\mathrm{d}^m}{\mathrm{d}x^m}\mathrm{P}_n(x)$$

其中 $\mathrm{P}_n(x)$**为勒让德多项式，其表达式如下：**

$$\mathrm{P}_n(x)=\frac{1}{2^n n!}\left[\frac{\mathrm{d}^n}{\mathrm{d}x^n}(x^2-1)^n\right]$$

有时需要对连带勒让德函数进行归一化，有施密特半归一化和完全归一化两种方法。施密特半归一化后的连带勒让德函数记为 $\mathrm{S}_n^m(x)$ **，它与** $\mathrm{P}_n^m(x)$ **的关系如下：**

$$\mathrm{S}_n^0(x)=\mathrm{P}_n^0(x)$$

$$S_n^m(x) = (-1)^m \sqrt{\frac{2(n-m)!}{(n+m)!}} P_n^m(x)(m>0)$$

完全归一化方法的目的是为了使归一化后的连带勒让德函数 $N_n^m(x)$ 满足下式：

$$\int_{-1}^{1} \left(N_n^m(x)\right)^2 dx = 1$$

完全归一化后的连带勒让德函数 $N_n^m(x)$ 与 $P_n^m(x)$ 的关系如下：

$$N_n^m(x) = (-1)^m \sqrt{\frac{\left(n+\frac{1}{2}\right)(n-m)!}{(n+m)!}} P_n^m(x)$$

在 MATLAB 7.0 中函数 legendre ()用于连带勒让德函数，其调用格式如下。

- *P* = legendre(*n*,*X*)，返回连带勒让德函数 $P_n^m(x)$，*m*=0,1,…,*n*，*n* 必须是标量整数，*x* 必须在[-1,1]范围内的实数。如果 ***X*** 是 1×*q* 的向量，则 ***P*** 是(*n*+1)×*q* 的矩阵，矩阵元素 *P*(*m*+1,*i*)代表 $P_n^m(x(i))$。一般来说，矩阵 ***P*** 总是比矩阵 ***X*** 多一个维度；
- *S* = legendre(*n*,*X*,'sch')，返回施密特半归一化后的连带勒让德函数 $S_n^m(x)$；
- *N* = legendre(*n*,*X*,'norm')，返回完全归一化后的连带勒让德函数 $N_n^m(x)$。

3.3.2 数论函数

MATLAB 7.0 提供的数论函数及其功能如表 3-11 所示。

表 3-11 数论函数

函数名	功能描述	函数名	功能描述
factor	分解质因子	rat	把实数近似为有理数
isprime	是否为素数	rats	利用 rat 函数来显示输出
primes	小于等于输入值的素数	perms	给出向量的所有置换
gcd	最大公因素	nchoosek	计算 C_n^k，即 *n* 种事物中一次取出 k 种事物的组合数目
lcm	最小公倍数	factorial	阶乘

例如，求 78 的所有质因子，具体代码设置如下：

```
f=factor(78)
```

由上述语句得到如下代码：

```
f =
      2      3     13
```

例如，求 C_{10}^{3}，具体代码设置如下：

```
c=nchoosek(10,3)
```

由上述语句得到如下代码：

```
c =
    120
```

3.3.3　坐标变换函数

MATLAB 7.0 提供的坐标变换函数及其功能如表 3-12 所示。

表 3-12　坐标变换函数

函数名	功能描述	函数名	功能描述
cart2sph	笛卡尔坐标系转换为球坐标系	sph2cart	球坐标系转换为笛卡尔坐标系
cart2pol	笛卡尔坐标系转换为极坐标系	hsv2rgb	灰度饱和度颜色空间转换为 RGB 颜色空间
pol2cart	极坐标系转换为笛卡尔坐标系	rgb2hsv	RGB 颜色空间转换为灰度饱和度颜色空间

例如，把笛卡尔坐标系中的点（1,1,1）分别转换到球坐标系和极坐标系中，代码设置如下：

```
[THETA,PHI,R] = cart2sph(1,1,1)
[THETA,RHO,Z] = cart2pol(1,1,1)
```

由上述语句得到如下代码：

```
THETA =
     0.7854
PHI =
     0.6155
R =
     1.7321
THETA =
     0.7854
RHO =
     1.4142
Z =
      1
```

第 4 章　MATLAB 7.0 基本编程

MATLAB 作为一种广泛应用于科学计算的工具软件，不仅具有强大的数值计算、符号计算、矩阵运算的能力和丰富的画图功能，还可以像 C 语言、FORTRAN 等计算机高级语言一样进行程序设计，编写扩展名为.m 的 M 文件，实现各种复杂的运算，这使得 MATLAB 在科研中的应用更加深入，常常作为系统仿真的工具应用。MATLAB 提供文件编辑器和编译器，这为用户带来了方便，事实上，MATLAB 自带的许多函数就是 M 文件函数，用户也可利用 M 文件来生成和扩充自己的函数库。

所谓 M 文件，简单来说就是用户把要实现的命令写在一个以.m 作为扩展名的文件中，然后由 MATLAB 系统进行解释，最后运行出结果。由此可见 MATLAB 具有强大的可开发性和可扩展性。另外，由于 MATLAB 是由 C 语言开发而成的，因此 M 文件的语法规则与 C 语言几乎一样，简单易学。

本章将讨论 MATLAB 中编程的规则和注意事项，并给出很多典型的例程帮助读者尽快熟悉。

4.1　脚本和函数

M 文件有函数（Functions）和脚本（Scripts）两种格式。二者相同之处在于它们都是以 m 作为扩展名的文本文件，不进入命令窗口，而是由文本编辑器来创建外部文本文件。但是两者在语法和使用上略有区别。

4.1.1　函数

MATLAB 中许多常用的函数（如 sqrt、inv 和 abs 等）都是函数式 M 文件，使用时，MATLAB 获取传递给它的变量，利用操作系统所给的输入，运算得到要求的结果，然后返回这些结果。函数文件类似于一个黑箱，由函数执行的命令以及这些命令所创建的中间变量都是隐含的。运算过程中的中间变量都是局部变量（除特别声明外），存放在函数本身的工作空间内，不会和 MATLAB 基本工作空间（Base workspace）的变量相互覆盖，对用户来说，可见的只是输入和输出，因此易于使程序模块化，特别适合于大型程序代码。

下面通过一个函数的例子来说明其结构。

```
>>type function1.m
function average=function1(value)
average=sum(value)/length(value);
```

此函数的第一行为函数定义行，以 function 语句作为引导，定义了函数名称（function1，需要注意的是函数名和文件名必须相同）、输入自变量（*value*，调用此函数时指定此变量的

值，类似于 C 语言的形式参数）和输出自变量（*average*，函数执行完毕返回的结果）。function 为关键词，说明此 M 文件为函数。第二行则为函数主体，规范函数的运算过程，并指出输出自变量的值。若调用此函数，可输入以下命令：

```
>>value=[1 2 3];
>>average=function1(value)
average=
        2
```

此外，还可在函数定义行下加入注解，以%开头，即函数的在线帮助，若在 MATLAB 输入“ help 函数主文件名 ”，即可看到这些帮助，需要注意的是，在线帮助和 M 函数定义行之间可以有空行，但是在线帮助的各行之间不应有空行。示例代码如下：

```
>>type function2.m
function average=function2(vector)
%function2 A simple function with a single help line
%usage of this function
%"output" is the average of the input vector "input".
%Roger Jang,19991123.
average=sum(vector)/length(vector);     %计算平均值
>>help function2
function2 A simple function with a single help line
"output" is the average of the input vector "input".
```

如果函数中有注释，代码设置如下：

```
>>type function2.m
function average=function2(vector)
%function2 A simple function with a single help line

%usage of this function
%"output" is the average of the input vector "input".
%Roger Jang,19991123.
average=sum(vector)/length(vector);     %计算平均值
```

则在线帮助只有第一行：

```
>>help function2
function2 A simple function with a single help line
```

以上的例子只是为了说明函数的性质和用法，有关编程的方法、具体问题以及典型例子会在后面的章节介绍。

4.1.2　脚本

脚本是一个扩展名为.m 的文件，其中包含了 MATLAB 的各种命令，与批处理文件很类似，在 MATLAB 命令窗口下直接输入此文件的主文件名，MATLAB 可逐一执行在此文件内的所有命令，和在命令窗口逐行输入这些命令一样。脚本式 M 文件运行产生的所有变量都是

全局变量，运行脚本后，所产生的所有变量都驻留在 MATLAB 基本工作空间内，只要用户不使用 clear 命令加以清除，且 MATLAB 指令窗口不关闭，这些变量将一直保存。基本空间随 MATLAB 的启动而产生，在关闭 MATLAB 软件时该基本空间被删除。

例如，假设当前目录下有一个脚本 M 文件，可用 type 命令显示其内容如下：

```
>>type solver.m
%solver.m
%used to solve A*X=b
%where A=[-1.5 1 2;3 -1 1;-1 3 5], b=[2.5;5;8].
A=[-1.5 1 2;3 -1 1;-1 3 5];
b=[2.5;5;8];
X=A\b
```

在上面的示例代码中以%开头的行是注释，在命令窗口执行 solver 命令，即可得到方程组的解，具体代码如下：

```
>>solver
X=
    0.7500
    -0.6250
    2.1250
```

结合上例，下面对 M 文件必须遵循的规则及两种类型的异同做简要介绍。

（1）从函数名必须与文件名相同。

（2）脚本式 M 文件没有输入参数或输出参数，而函数式 M 文件有输入参数和输出参数。

（3）函数可以有零个或多个输入和输出变量。函数 nargin 和 nargout 包含输入和输出变量的个数。在运行时，可以按少于 M 文件中规定的输入和输出变量的个数进行函数调用，但不能多于这个标称值。

从运行上看，与脚本文件不同的是，函数文件被调用时，MATLAB 会专门为它开辟一个临时工作空间，称为函数工作空间（Function workspace），用来存放中间变量，当执行完函数文件的最后一条命令或者遇到 return 时就结束该函数文件的运行，同时该临时函数空间及其所有的中间变量将被清除。函数工作空间相对于基本空间是临时的、独立的，在 MATLAB 运行期间，可以产生任意多个临时函数空间。

（4）在 M 文件中，包括脚本和函数，到第一个非注释行为止的注释行是帮助文本，当需要帮助时，返回该文本，通常用来说明文件的功能和用法。

（5）函数 M 文件中的所有变量除特殊声明外都是局部变量，而脚本中的变量都是全局变量。

（6）变量的命名可以包括字母、数字和下划线，但必须是以字母开头。并且在 M 文件设计中是区分大小写的。变量的长度不能超过系统函数 namelengthmax 所规定的值。

（7）假设在函数文件中发生对某脚本文件的调用，那么该脚本文件运行产生的所有变量都存放于此函数空间中，而不是存在基本工作空间中。

通常 M 文件是文本文件，所以可使用一般的文本编辑器编辑 M 文件，存储时以文本模式存储。此外，MATLAB 内部自带了 M 文件编辑器与编译器，可选择 MATLAB 命令窗口上的 file/new 子菜单，然后选择下一级子菜单中的 M-file 子菜单，就进入了 M 文件编辑/编译器，

如图 4-1 所示。它是一个集编辑与调试两种功能于一体的工具环境。进行代码编辑时，它可以用不同的颜色来显示注解、关键词、字符串和一般程序代码，使用非常方便。在书写完 M 文件后，也可以像一般的程序设计语言一样，对 M 文件进行调试、运行。

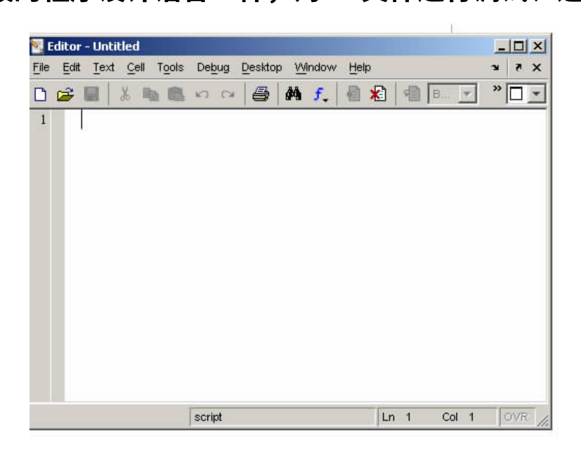


图 4-1　M 文件编辑器和编译器

4.1.3　子函数与私有目录

一个 M 文件可以包含一个以上的函数，其中有一个主函数，其他为子函数。这些子函数只能被同一文件中的函数（主函数或其他子函数）调用，但是不能被其他文件的函数调用。在一个 M 文件中，主函数必须出现在最上方，其后可接上任意数目的子函数，而且子函数的次序可随意。同一文件的主函数、子函数的工作空间都是彼此独立的，各函数间的信息可通过输入输出宗量、全局变量或跨空间指令传递。

此外，可以在某一目录中建立一个自己命名的私有目录来存放相关的函数，例如在 work 目录下建立一个 rscode 目录，则 work 中的 M 文件（无论是脚本还是函数）即可调用 rscode 下的任何函数，而不必再定义其他搜寻路径。在 rscode 下的函数，只能被其父目录的函数所调用而不能被其他目录下的函数调用。

当 M 文件中需要调用某一个函数时，MATLAB 是按照以下顺序来搜寻的：

- 检查此函数是否是子函数；
- 检查此函数是否为私有目录的函数；
- 从所设定的搜寻路径搜索此函数。

搜索过程中，只要找到与第一个文件名相符的函数就会立即取用而停止搜索。

4.1.4　P 码文件

P 码是伪代码 Psedocode 的简写，一个 M 文件首次被调用时，MATLAB 将首先对该 M 文件进行语法分析，并把生成的相应内部伪代码（P 码）存放在内存中。此后当再次调用该 M 文件时，将直接运行该文件在内存中的 P 码文件而不会对原码文件重复进行语法分析。P 码文件和原码文件具有相同的文件名，但其扩展名为“.p”。并且其运行速度要高于原码文件，但是对于规模不大的文件，用户一般察觉不到这种速度上的优势。

尤其在 MATLAB 环境中，假如存在同名的 P 码和原码文件，则当该文件名被调用时，被执行的肯定是 P 码文件。

P 码文件不是只有当 M 文件被调用时才可产生，也可被预先生成。这种功能可以用作代码保护的手段，生成 P 码文件后，其他用户可以使用该代码，但是无法看到代码的内容。具体操作如下：

```
pcode filename    %在当前目录下生成 filename.p
pcode filename – inplace    %在 filename.m 所在目录下生成 filename.p
```

如果要在内存中对 P 码文件进行操作，可键入以下命令：

```
inmem       %罗列出内存中的所有 P 码文件
clear filename      %清除内存中的 filename.pP 码文件
clear functions      %清除内存中的所有 P 码文件
```

4.2 MATLAB 中的变量和语句

MATLAB 能识别一般常用的加、减、乘、除和幂等运算，对于简单的计算可以在命令窗口中输入表达式后，按 Enter 键即可完成。MATLAB 会将运算结果存入默认变量 *ans* 中，并显示其结果（如果表达式后加上分号“；”，则不会显示），当然用户也可设定自己的变量。与 C 语言不同的是，MATLAB 中的变量是不需要事先定义的。

MATLAB 的主要功能虽然是数值运算，但是它也是一个完整的程序语言，有各种语句格式和语法规则。下面就进行详细介绍。

4.2.1 变量类型

与 C 语言的区别是，在用 M 语言编写程序的过程中，变量不需要事先定义。但 MATLAB 中的变量也有自己的命名规则，即必须以字母开头，之后可以是任意字母、数字或下划线，不能有空格；变量名区分大小写；在 MATLAB 7.0 中，变量名不能超过 63 个字符，第 63 个字符之后的部分将被忽略。

除了上述命名规则外，MATLAB 还包括一些特殊的变量，如表 4-1 所示。

表 4-1　　MATLAB 中的特殊变量

变量名称	变量含义
ans	MATLAB 中默认变量
pi	圆周率
eps	计算机中的最小数
inf	无穷大
NaN	不定值，如 0/0
i(j)	复数中的虚数单位
nargin	所用函数的输入变量数目
nargout	所用函数的输出变量数目
realmin	最小可用正实数
realmax	最大可用正实数

用户定义的变量有局部变量和全局变量两种类型。每一个函数在运行时，均占用单独的

一块内存，此工作空间独立于 MATLAB 的基本工作空间和其他函数的工作空间，因此不同工作空间的变量完全独立不会相互影响，这些变量称为局部变量。有时为了减少变量的传递，可使用全局变量，它是通过 global 指令定义的，格式为：

```
global var1 var2;
```

通过上述指令，可以使 MATLAB 允许几个不同的函数空间以及基本工作空间共享同一个变量。每个希望共享全局变量的函数或 MATLAB 基本工作空间必须逐个对具体变量加以专门定义，没有采用 global 定义的函数或基本空间将无权使用全局变量。

如果某个函数的运行使得全局变量发生了变化，则其他函数空间及基本工作空间内的同名变量随之变化。只要与全局变量相联系的工作空间有一个存在，则全局变量存在。

在使用全局变量中需要注意以下几个方面。

- 在使用之前必须首先定义，建议将定义放在函数体的首行位置。
- 虽然对全局变量的名称并没有特别的限制，但是为了提高程序的可读性，建议采用大写字符命名全局变量。
- 全局变量会损坏函数的独立性，使程序的书写和维护变得困难，尤其是在大型程序中，不利于模块化，这里不推荐使用。

例如，在函数 sum2(y)中声明了一个全局变量，内容如下：

```
function z=sum2(y)
global X
z=X+y;
```

要测试此函数，可在命令窗口进行如下操作：

```
>>global X        %在基本工作空间进行全局变量 X 的声明
>>X=3
>> z=sum2(2)
z =
      5
>> whos global                %查看工作空间的全局变量
   Name          Size                              Bytes    Class
   X              1x1                                     8    double array (global)
Grand total is 1 element using 8 bytes
clear global X            %清除全局变量 X
```

除命名规则外，变量命名时还需要注意以下两个方面：

（1）不要把函数名用作变量，否则在没有从内存中清除该变量的情况下将不能调用该函数，要测试变量名是否已用作函数名，可键入 which –all<name>进行确认；

（2）MATLAB 预留了一些关键字并且不允许重载它们，定义变量时要避开这些关键字，（用 iskeyword 可列出所有的预留关键字）否则系统会显示类似于缺少操作数之类的错误信息。

4.2.2　M 文件的流控制语句

一般来讲，决定程序结构的语句可分为顺序语句、循环语句和分支语句 3 种，每种语句有各自的流控制机制，相互配合使用可以实现功能强大的程序。

1．顺序语句

顺序语句就是依次顺序执行程序的各条语句，批处理文件就是典型的顺序语句的文件，这种语句不需要任何特殊的流控制。示例代码如下：

```
%定义变量 a
a=[15,20,25,30,35];
%定义变量 b
b=[1554.88,1555.24,1555.76,1556.20,1556.68];
figure(1)
%使用缺省设置进行作图，以 a 为横轴，b 为纵轴
plot(a,b)
hold on
%用红色标志'x'画出相关的点
plot(a,b,'rx');
```

2．循环语句

循环语句一般用于有规律的重复计算。被重复执行的语句称为循环体，控制循环语句走向的语句称为循环条件。MATLAB 中有 for 循环和 while 循环两种语句。

（1）for 循环

for 循环的语法结构如下：

```
for  循环变量=数组
    循环体；
end
```

可以看出，这种语句与其他语言中的 for 循环结构是相同的，循环体的执行次数是确定的，它是由数组的列数决定。此外，for 循环是可以多重嵌套的，请看下面的示例代码：

```
%根据中心极限定理，从双极性分布的序列中获得满足正态分布的随机序列
clear;
n=50;
p=0.5;
%生成一个满足双极性分布的矩阵，大小为 500*1000
x=binornd(n,p,500,1000);
z=(x-25)/sqrt(12.5);
for i=1:1:1000
      y(1,i)=0;
end
for i=1:1:1000
      for j=1:1:500
      %将独立分布的双极性随机数相加
y(1,i)=y(1,i)+z(j,i);
end
```

```
%对最终的数据进行调整
y(1,i)=y(1,i)/sqrt(500);
end
```

注意：不能在 for 循环体内重新对循环变量赋值来终止循环的执行，有专门的命令 break 可以完成这一功能，后面会具体介绍。为得到高效代码，应尽可能提高代码的向量化程度，采用矩阵运算，而避免使用循环结构，如果使用的话，在循环指令之前尽量对数组进行预定义。

（2）while 循环

while 循环的语法结构如下：

```
while  表达式
        循环体；
end
```

while 循环的次数是不固定的，只要表达式的值为真，循环体就会被执行。通常表达式给出的是一个标量值，但也可以是数组或者矩阵，如果是后者，则要求所有的元素都必须为真。示例代码如下：

```
t=zeros(1,6);
i=1;5
while i<=6
      t(1)=1/i;
      i=i+1;
end
```

3．条件语句

在程序中如果需要根据一定条件来执行不同的操作时就需要用到条件语句了，MATLAB 中有 if-else-end 语句和 switch-case-otherwise 语句两种条件语句。

（1）if-else-end 语句

其语法结构如下：

```
if  条件式
   表达式 1；
else
   表达式 2；
end
```

也可有更简化的结构：

```
if  条件式
   表达式；
end
```

有多个条件式的复杂结构：

```
if  条件式 1
   表达式 1；
```

```
elseif 条件式2
  表达式2；
elseif 条件式3
  表达式3；
……
else
  表达式n；
end
```

如果在条件式中使用矩阵，则必须矩阵元素都不为0时，条件式才算成立。

下面是一个简单的分支语句的例程，代码设置如下：

```
function y=control(n)
a=20;
if n==0
    y=a+1;
elseif n==1
    y=a*(1+n);
elseif n==2
    y=a+n;
else
    y=a;
end
```

（2）switch-case-otherwise 语句

此语句与C语言中的选择语句是相同功能的，它通常用于条件较多而且较单一的情况，类似于一个数控的多路开关。其语法结构如下：

```
switch expression
  case value1
      statements1;
  case value2
      statements2;
      ……
  case valuen
      statementsn;
  otherwise
      statements;
end
```

expression 是一个标量或者字符串，将 expression 的值依次和各个 case 指令后面的检测值进行比较，当比较结果为真时，MATLAB 执行后面的一组命令，然后跳出该 switch 结构。如果所有的比较结果都为假，则执行 otherwise 后的命令。当然 otherwise 指令也可以不存在。

下面是 MATLAB 工具箱中的 itemsmsg 函数的示例，其中运用了 switch 结构，代码设置

如下：

```
function str = itemsmsg(type, items)
str = '';
switch type
    case 'Error'
        % Display each item's error message
        errItem = {items.ItemID};
        errMsg = {items.ErrorMessage};
        allStrs = [errItem; errMsg];
        str = sprintf('\t\t%s returned: "%s"\n', allStrs{:});
    case 'ReadAsync'
        str = sprintf('\t%d items read.', length(items));
    case 'WriteAsync'
        str = sprintf('\t%d items written.', length(items));
end
```

4．其他流控制语句

在许多程序设计中会碰到需要提前终止循环、跳出子程序、显示出错信息等情况，因此还需要其他的流控制语句来实现这些功能。主要有 continue、break、return、echo、error、try…catch 等。下面对各语句分别进行介绍。

（1）continue

此命令的作用是结束本次循环，即跳过循环体中尚未执行的语句，接着进行下一次是否执行循环的判断。以下是 MATLAB 自带的 M 文件 magic.m 中运用 continue 的示例，代码设置如下：

```
fid=fopen('magic.m','r');
count=0;
while ~feof(fid)
    line=fgetl(fid);
    if isempty(line)|strncmp(line,'%',1)
        continue
    end
    count=count+1;
end
disp(sprintf('%d lines',count));
```

（2）break

此命令的作用是终止本次循环，跳出最内层循环，即不必等到循环的结束而是根据条件退出循环，它的用法和 continue 类似，常常和 if 语句合用来强制终止循环。示例代码如下：

```
fid=fopen(' fft.m', 'r');
s='';
```

```
while ‘feof(fid)
    line=fgetl(fid);
    if isempty(line)
        break
    end
    s=strvcat(s,line);
end
disp(s);
```

需要注意的是，当 break 命令碰到空行时，将退出 while 循环。

（3）return

此命令可使正在运行的函数正常退出，并返回调用它的函数继续运行，经常用于函数的末尾以正常结束函数的运行，当然也可用在某条件满足时强行结束执行该函数。例如在 showopcevents.m 函数中，就运用了这一控制命令。

```
if isempty(eventStruct),
    out = [];
    return;
end
```

（4）echo

通常执行 M 文件时，在命令窗口是看不到执行过程的，但在特殊情况下比如需要作演示，要求 M 文件的每条命令都要显示出来，可以用 echo 命令实现这样的操作。

对于脚本式 M 文件和函数式 M 文件，echo 命令有所不同，对于脚本式 M 文件，echo 命令可以用以下方式来实现：

```
echo on      %显示其后所有执行的命令文件的指令
echo off     %不显示其后所有执行的命令文件的指令
echo         %在上述两种情况之间切换
```

对于函数式 M 文件，echo 命令可以用以下方式来实现：

```
echo filename on       %使 filename 指定的 M 文件的执行命令显示出来
echo filename off      %使 filename 指定的 M 文件的执行命令不显示出来
echo on all            %其后的所有 M 文件的执行指令显示出来
echo off all           %其后的所有 M 文件的执行指令显示出来
```

（5）error

此指令是用来指示出错信息并终止当前函数的运行。语法格式如下：

```
error(‘message’)
```

类似的还有 warning 指令，二者区别在于 warning 指示警告信息后程序仍继续运行。

（6）try…catch

其功能与 error 类似，用于对异常情况进行处理，其语法结构如下：

```
try
  (commands1)
catch
```

```
    (commands2)
end
```

组命令 1 总会被执行，当执行出现错误时，catch 控制块就可捕获它，执行组命令 2，针对不同的错误类型进行不同的处理。可调用 lasterr 函数查询出错原因。示例代码如下：

```
clear;
n=4;
a=magic(3);
try
    a_n=a(n,:),
catch
    a_end=a(end,:),
end
lasterr
```

运行结果如下：

```
a_end =
     4     9     2
ans =
Attempted to access a(4,:); index out of bounds because size(a)=[3,3].
```

注意 try 和 catch 控制块中的语句之间用逗号隔开。

（7）input

此命令用来提示用户从键盘输入数据、字符串或表达式，并接收输入值。语法格式如下：

```
user_entry=input('prompt')    %在屏幕上显示提示信息 prompt，等待用户的输入，并将输入赋给变量 user_entry
user_entry=input('prompt','s')    %返回的字符串作为文本变量而不是作为变量名或者数值
```

下面是一个等待用户确认显示的 M 文件的示例，代码设置如下：

```
function test()
r=input('Do you want more?Y/N[Y]:','s');
if isempty(r)
    r='Y';
end
if r=='Y'
    disp('you have selected the first character');
else
    disp('you have selected the second one');
end
```

用户运行这个文件时，命令窗口将出现以下信息：

```
>>test
Do you want more?Y/N[Y]:
%按回车键表示确认
```

```
you have selected the first character
>>test
Do you want more?Y/N[Y]: n
%按 n 键，然后按回车键
you have selected the second one
```

（8）keyboard

此命令被放置在 M 文件中，将停止文件的执行并将控制权交给键盘。通过在提示符前显示 K 来表示一种特殊状态。在 M 文件中使用该命令，对程序的调试和在程序运行中修改变量都很方便。

如果在上例中某个位置加入 keyboard 命令，则执行到这句话时，MATLAB 的命令窗口将显示如下代码：

```
>> test
K>>
```

（9）pause

此命令用于暂时中止程序的运行，等待用户按任意键继续进行。该命令在程序的调试过程和用户需要查询中间结果时使用很方便。该命令的语法格式如下：

```
pause      %停止 M 文件的执行，按任意键继续
pause(n)   %中止执行程序 n 秒后继续，n 是任意实数
pause on   %允许后续的 pause 命令中止程序的运行
pause off  %禁止后续的 pause 命令中止程序的运行
```

4.3 程序的调试（Debug）

对于编程者来说，程序运行时出现 bug 在所难免，尤其是在大规模、多人共同参与的情况下，因此掌握程序调试的方法和技巧对提高工作效率很重要。一般来说，错误可分为两种，即语法错误（Syntax Errors）和逻辑错误（Logic Errors）。语法错误一般是指变量名与函数名的误写、标点符号的缺漏和 end 的漏写等，对于这类错误，MATLAB 在运行或 P 码编译时一般都能发现，终止执行并报错，用户很容易发现并改正。而逻辑错误可能是程序本身的算法问题，也可能是用户对 MATLAB 的指令使用不当，导致最终获得的结果与预期值偏离，这种错误发生在运行过程中，影响因素比较多，而这时函数的工作空间已被删除，调试起来比较困难。

下面针对上述的两种错误推荐两种调试方法，即直接调试法和工具调试法。

4.3.1 直接调试法

MATLAB 本身的运算能力强，指令系统比较简单，因此程序一般都显得比较简洁，对于简单的程序采用直接调试法往往还是很有效的。通常采取的措施如下。

（1）通过分析后，将重点怀疑语句后的分号删掉，将结果显示出来，然后与预期值进行

比较。

（2）单独调试一个函数时，将第一行的函数声明注释掉，并定义输入变量的值，然后以脚本方式执行此 M 文件，这样就可保存下原来的中间变量了，可以对这些结果进行分析，找出错误。

（3）可以在适当的位置添加输出变量值的语句。

（4）在程序中的适当位置添加 keyboard 指令。当 MATLAB 执行至此处时将暂停，并显示 k>>提示符，用户可以查看或改变各个工作空间中存放的变量，在提示符后键入 return 指令可以继续执行原文件。

但是对于文件规模大，相互调用关系复杂的程序，直接调试是很困难的，这时可以借助于 MATLAB 的专门工具调试器（Debugger）进行，即工具调试法。

4.3.2　工具调试法

MATLAB 自身包括调试程序的工具，利用这些工具可以提高编程的效率，包括一些命令行形式的调试函数和图形界面形式的菜单命令。实际工作中，可以根据个人需要进行操作。本节主要介绍一些基本方法，在这些方法的基础上还需要读者不断实践，总结经验，才能做到熟练运用，高效编程。

1．以命令行为主的程序调试

以命令行为主的程序调试手段具有通用性，可以适用于各种不同的平台，它主要是应用 MATLAB 提供的调试命令。在命令窗口输入 help debug 可以看到一个对于这些命令的简单描述，下面分别进行介绍。

（1）设置断点

这是其中一个最重要的部分，可以利用它来指定程序代码的断点，使得 MATLAB 可在断点前停止执行，从而可以检查各个局部变量的值。函数格式有以下几种：

- dbstop in mfile

在文件名为 mfile 的 M 文件的第一个可执行语句前设置断点，执行该命令后，当程序运行到 mfile 的第一个可执行语句时，可暂时中止 M 文件的执行，并进入 MATLAB 的调试模式。M 文件必须处在 MATLAB 搜索路径或当前目录内。如果用户已经激活了图形调试模式，则 MATLAB 调试器将打开该 M 文件，并在第一个可执行语句前设置断点。

- dbstop in mfile at lineno

在文件名为 mfile 的 M 文件的第 lineno 行设置断点，执行过程与上一命令类似。如果行号为 lineno 的语句为非执行语句，则停止执行的同时，在该行号的下一个可执行语句前设置断点。M 文件必须处在 MATLAB 搜索路径或当前目录内。此时，用户可以使用各种调试工具、查看工作空间变量、公布任何有效的 MATLAB 函数。

- dbstop in mfile at subfun

执行该命令后，当程序执行到子程序 subfun 时，暂时中止文件的执行并使 MATLAB 处于调试模式，其他要求和操作与上面的函数类似。

- dbstop if error

执行该命令后，可在运行 M 文件遇到错误时，终止 M 文件的执行，并使 MATLAB 处于调试状态，运行停止在产生错误的行。这里的错误不包括 try…catch 语句中检测到的错误，用户不能在错误后重新开始程序的运行。

- dbstop if all error

与上一命令类似，但是在执行该命令时遇到任何类型的运行错误时均停止，包括在 try…catch 语句中检测到的错误。

- dbstop if warning

执行该命令后，在运行 M 文件遇到警告时，终止 M 文件的执行，并使 MATLAB 处于调试状态，运行将在产生警告的行暂停，程序可以恢复运行。

- dbstop if caught error

执行该命令后，当 try…catch 检测到运行时间错误时，停止 M 文件的执行，用户可以恢复程序的运行。

- dbstop if naninf 或 dbstop if infnan

执行该命令后，当遇到无穷值或者非数值时，终止 M 文件的执行。

下面给出一个在命令窗口设置断点的示例。

运行的 M 文件内容如下：

```
function y=test1(x)
l=length(x);
y=(1:l)+x;
```

在命令窗口输入以下内容：

```
>> dbstop in test1.m
>> test1(1:10)
```

在打开的 M 文件窗口中设置断点的情况如图 4-2 所示，在第一行设置了一个断点。

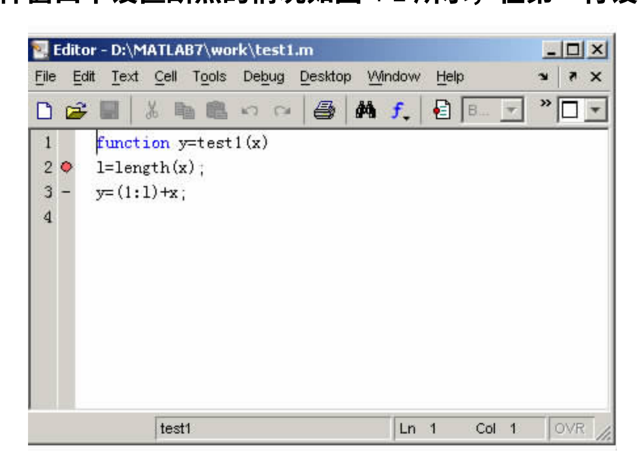


图 4-2　断点图示

test1 函数中输入只能为向量，如果输入矩阵，会产生错误。因此可以根据错误进入 MATLAB 调试状态，在命令窗口输入如下：

```
>> dbstop if error
>> test1(magic(3))
```

```
??? Error using ==> unknown
Matrix dimensions must agree.
Error in ==> test1 at 3
y=(1:l)+x;
K>>
```

此时，M 文件执行到最后一行，如图 4-3 所示。

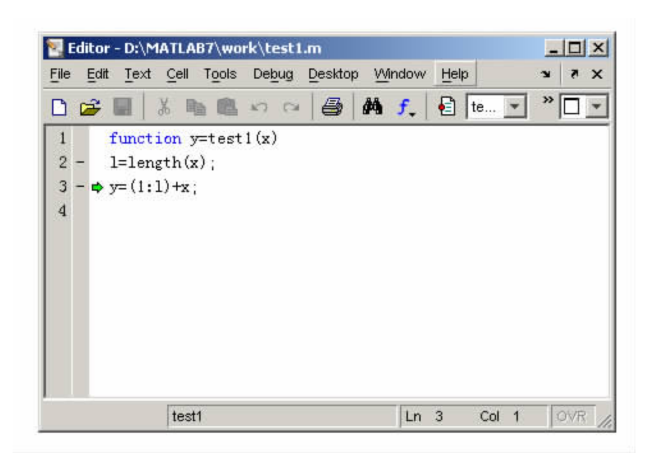


图 4-3　文件执行情况图示

由上例可以看到，程序停止执行后，MATLAB 进入调试模式，命令行上出现 K>>的提示号，代表此时可以接受键盘输入。

（2）清除断点

- dbclear all

清除所有 M 文件中的所有断点。

- dbclear all in mfile

清除文件名为 mfile 的 M 文件中的所有断点。

- dbclear in mfile

清除 mfile 中第一个可执行语句前的断点。

- dbclear in mfile at lineno

清除 mfile 中行号为 lineno 的语句前的断点。

- dbclear in mfile at subfun

清除 mfile 中子函数 subfun 行前的断点。

- dbclear if error

清除由 dbstop if error 设置的暂停断点。

- dbclear if warning

清除由 dbstop if warning 设置的暂停断点。

- dbclear if naninf

清除由 dbstop if naninf 设置的暂停断点。

- dbclear if infnan

清除由 dbstop if infnan 设置的暂停断点。

（3）恢复执行

- dbcont

从断点处恢复程序的执行，直到遇到程序的另一个断点或错误后返回 MATLAB 基本工作空间。

（4）调用堆栈

- dbstack

此命令显示 M 文件名和断点产生的行号，调用此 M 文件的名称和行号等，直到最高级 M 文件函数，即列出了函数调用的堆栈。

使用此命令时，有如下格式：

```
[ST,I] = dbstack
```

通过 M×1 的结构体 ST 形式返回堆栈信息。ST 的形式有以下几种。

file —— 函数出现的文件名，如果没有则为空；

name —— 文件中的函数名；

line —— 函数行号。

I 是当前的工作空间的索引。

- dbstack(N)

此命令省略了显示中的前 *N* 个帧。

- dbstack ('-completenames')

此命令输出堆栈中的每个函数的全名，即函数文件的名称和在堆栈中函数包含的关系。

（5）列出所有断点

- dbstatus

此命令列出所有的断点，包括错误、警告、nan 和 inf 等。

s=dbstatus 将通过一个 M×1 的结构体来返回断点信息，结构体中有以下字段。

name —— 函数名；

line —— 断点行号向量；

expression —— 与 line 中相对应的断点条件表达字符串；

cond —— 条件字符串，如 error、caught error、warning 或 naninf；

identifier —— 当条件字符串是 error、caught error 或 warning 时，该字段是 MATLAB 的信息指示字符串。

- dbstatus mfile

此命令列出指定的 M 文件中的所有断点设置，mfile 必须是 M 文件函数的名称或者是 MATLAB 有效的路径名。

（6）执行 1 行或多行语句

- dbstep

执行当前 M 文件下一个可执行语句。

- dbstep nlines

执行下 nlines 行可执行语句。

- dbstep in

当执行下一个可执行语句时，如果其中包含对另外一个函数的调用，此命令将从被调用

的函数文件的第一个可执行语句执行。

- dbstep out

此命令将执行函数剩余的部分，在离开函数时停止。

这 4 种形式的语句执行完后，都返回调试模式，如果在执行过程中遇到断点，程序将中止。

（7）列出文件内容

- dbtype mfile

列出 mfile 文件的内容，并在每行语句前面加上标号以方便使用者设定断点。

- dbtype mfile start:end

列出 mfile 文件中指定行号范围的部分。

在 UNIX 和 VMS 调试模式下，并不显示 MATLAB 的调试器，此时必须使用 dbtype 来显示源程序代码。

（8）切换工组空间

- dbdown

遇到断点时，将当前工作空间切换到被调用的 M 文件的空间。

- dbup

将当前工作空间（断点处）切换到调用 M 文件的工作空间。两个命令常常配合使用。

（9）退出调试模式

- dbquit

立即结束调试器并返回到基本工作空间，所有断点仍有效。

2．以图形界面为主的程序调试

MATLAB 自带的 M 文件编辑器同时也是程序的编译器，用户可以在编辑完程序后直接进行调试，更加方便和直观。

新建一个 M 文件，即可打开编译器，选择主菜单中“Debug”选项，打开下拉菜单，有各种调试命令，如图 4-4 所示。

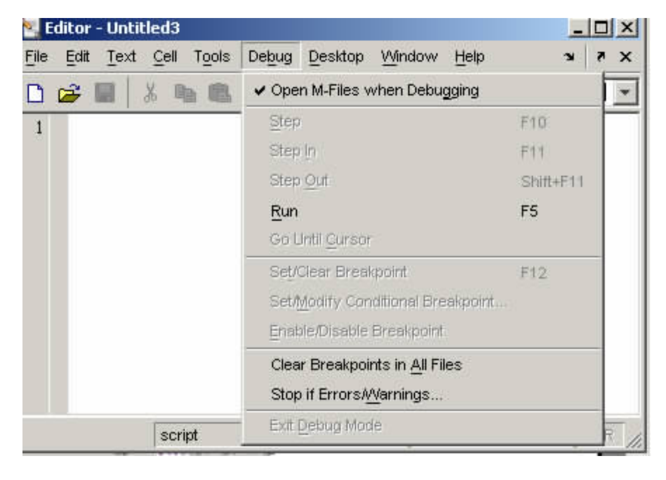


图 4-4　打开的 MATLAB 调试器图形界面

下拉菜单中的命令有一部分在工具栏中有图标相对应，其功能与上一节介绍的调试命令是相同的，下面只对各命令做简单介绍。

- step （）

单步执行，快捷键为 F10，与调试命令中的 dbstep 相对应。

- step in （）

深入被调函数，快捷键为 F11，与调试命令中的 dbstep in 相对应。

- step out （）

跳出被调函数，快捷键为 Shift+F11，与调试命令中的 dbstep out 相对应。

- run/continue （）

连续执行，快捷键为 F5，与调试命令中的 dbcont 相对应。

- go until cursor

运行到鼠标所在的行，与 dbstop in mfile at lineno 相对应。

- set/clear breakpoint （）

设置或清除断点，快捷键为 F12，与 dbstop 和 dbclear 相对应。

- set/modify conditional breakpoint…

设置或者修改条件断点，单击此菜单项时，会弹出如图 4-5 所示的对话框，要求用户对断点的条件作出设置，设置前光标在哪一行，则设置的断点就在这一行前。

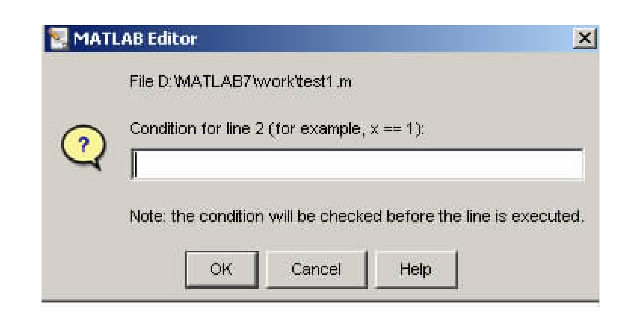


图 4-5　set/modify conditional breakpoint 对话框

- enable/disable breakpoint

允许或者禁止断点的功用。

- clear breakpoints in all files （）

清除所有断点，与 dbclear all 相对应。

- stop if errors/warnings

与 dbstop if error、dbstop if all error、dbstop if warning、dbstop if caught error、dbstop if naninf 和 dbstop if infnan 等命令等价，单击此菜单项时，将弹出一个对话框，如图 4-6 所示。

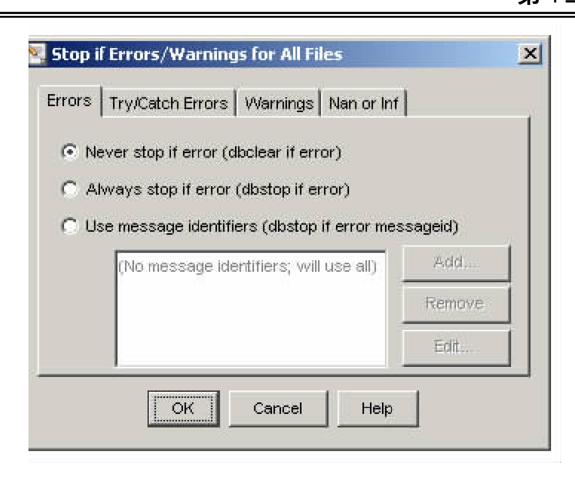


图 4-6　“Stop if Errors/Warnings for All Files”对话框

可以看到，对话框的每一栏都和一个调试命令相对应，用户可以在调试前根据自己的要求设定，然后运行程序。

- exit debug mode（）

退出调试模式，与 dbquit 相对应。

只有当文件进入调试状态时，上述命令才会全部处于使能态。需要注意的是，在调试器的工具栏的右侧，还有一个如图 4-7 所示的堆栈下拉菜单。在调试中，可以通过改变它的内容来观察和操作不同工作空间中的量，类似于调试命令中的 dbdown 和 dbup。

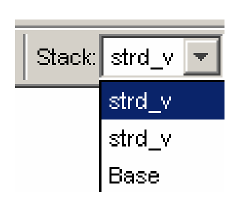


图 4-7　工作空间切换项

下面通过一个实例来说明一下这些菜单项的用法。

【例 4.1】计算向量的标准差。

函数文件名为 strd_v.m，具体代码如下：

```
function f=strd_v(x)
l=length(x);
s=sum(x);
y=s/l;
t=strd_fun(x,y);
f=sqrt(t/(l-1));
```

其中调用的子函数代码如下：

```
function f=strd_fun(x,y)
t=0;
for i=1:length(x)
      t=t+((x-y).^2);
```

```
end
f=t;
```

在命令窗口输入如下代码：

```
>> v=[1,2,3,4,5,6];
>> strd_v(v)
```

由上述语句得到如下结果：

```
ans =
    2.7386    1.6432    0.5477    0.5477    1.6432    2.7386
```

而 MATLAB 提供的同样功能的函数 *std* 计算结果如下：

```
>> std(v)
ans =
    1.8708
```

上面的示例说明函数的逻辑有问题，需要调试，具体操作步骤如下。

在文件 strd_v.m 最后一行前设置一个断点，编译器用一个大红点标记，如图 4-8 所示，程序运行时，将在断点处暂停。

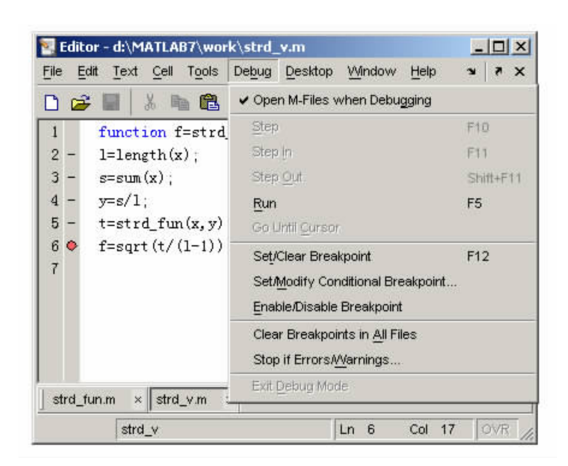


图 4-8 设置断点

运行程序，检查变量，代码设置如下：

```
>> strd_v(v)
K>>
```

当运行到断点处时，在断点和文本之间将会出现一个绿色箭头，表示程序运行至此停止，如图 4-9 所示。

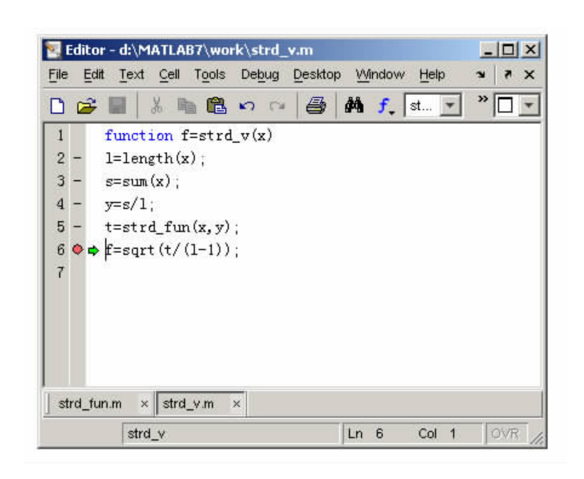


图 4-9　设置断点后程序的运行

在 K>>后检查变量 *l*,*s*,*y*,*t* 的数值，发现 *t* 有错误，结果如下：

```
K>> t
t =
    37.5000    13.5000    1.5000    1.5000    13.5000    37.5000
```

检查变量也可通过变量浏览器进行，如图 4-10 所示。双击变量，即可打开数组编辑器，可以查看和修改变量内容，如图 4-11 所示。由此可以确定断点以前的部分是正确的，出错的是 strd_fun 子程序，需对子程序进行调试。

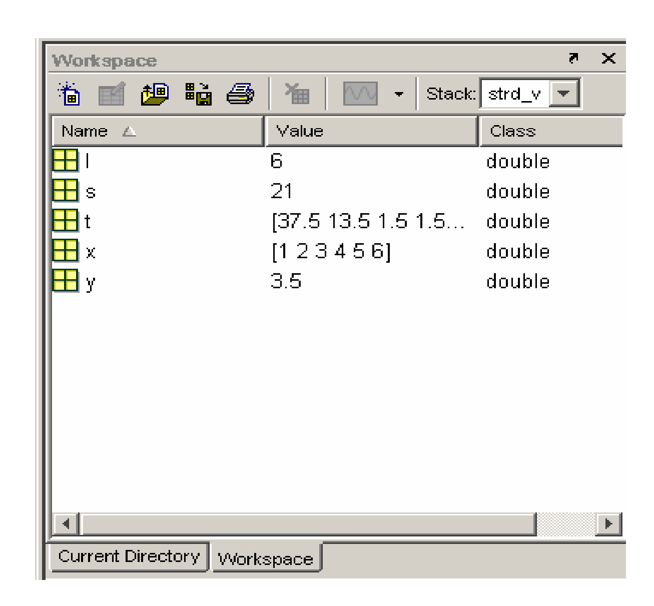


图 4-10　变量浏览器

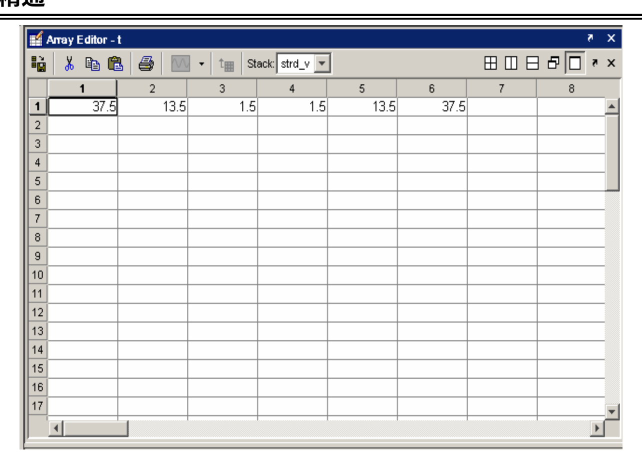


图 4-11　数组编辑器

切换工作空间到基本工作空间，即将图 4-7 中的“Stack”项选为“Base”。清除前面设置的断点，可以选择“Debug”下拉菜单中的“set/clear breakpoints”项，也可直接在大红标记上单击一下鼠标左键。此时，标记将会消除，绿色箭头变为白色。为了去除白色箭头，需单击“continue”按钮使程序继续运行。

然后调试子函数。打开 strd_fun.m 文件，在第 5 行设置一个断点，采用前面的向量 v 再一次调用函数 strd_v,这时单击函数文件 strd_v.m 的标签或者在 stack 下拉列表中选择 strd_v，则在文件的第 5 行前增加了一个白色的箭头，如图 4-12 所示，表示在主程序中的断点位置。查看变量值为：

```
K>> i
i =
     1
K>> t
t =
    6.2500    2.2500    0.2500    0.2500    2.2500    6.2500
```

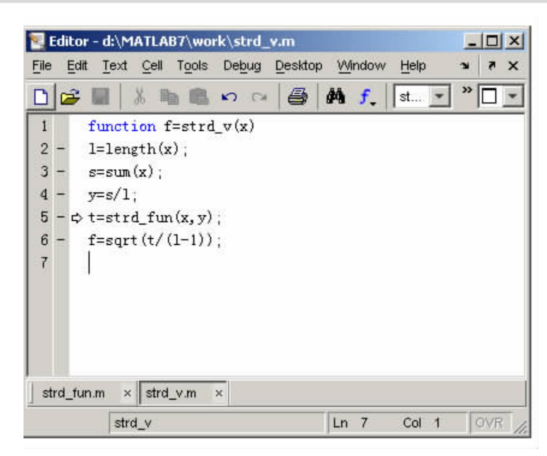


图 4-12　主程序的断点设置情况

可见，错误出在 t 的计算式上，观察发现 t=t+((x-y).^2)中的 x 应改为 x(i)。

退出调试后修改程序，清除断点，重新运行后得到正确的结果，代码设置如下：

```
>> strd_v(v)
ans =
      1.8708
```

本例只是为了说明一些命令的用法，实际中遇到的问题可能会更复杂，要求调试者必须要有耐心和细心。

4.4　函数的设计和实现

前面介绍了 MATLAB 的语法规则和使用方法，本节将通过一个示例具体来讲述如何用 MATLAB 来解决问题。

【例 4.2】用 MATLAB 对基带数字通信系统进行仿真。通信系统结构如图 4-13 所示，整个系统模拟了发送和接收的全过程。其中调制部分采用 16QAM，并且采用基带传输，不必调制到载频上（在实际操作中通信系统应该对基带信号进行载波调制，可以减少传输中的损耗，并且提高信道频带的利用率）。该系统采用了高斯信道，在接收端进行解调并与发送信号比较得到误码率。图 4-13 中还要求得到各个模块的输出结果图示。此外，在发送端的低通滤波后，有一个调制到载波的步骤，主要是为了观察实际系统中的波形，这是相对独立的一部分。

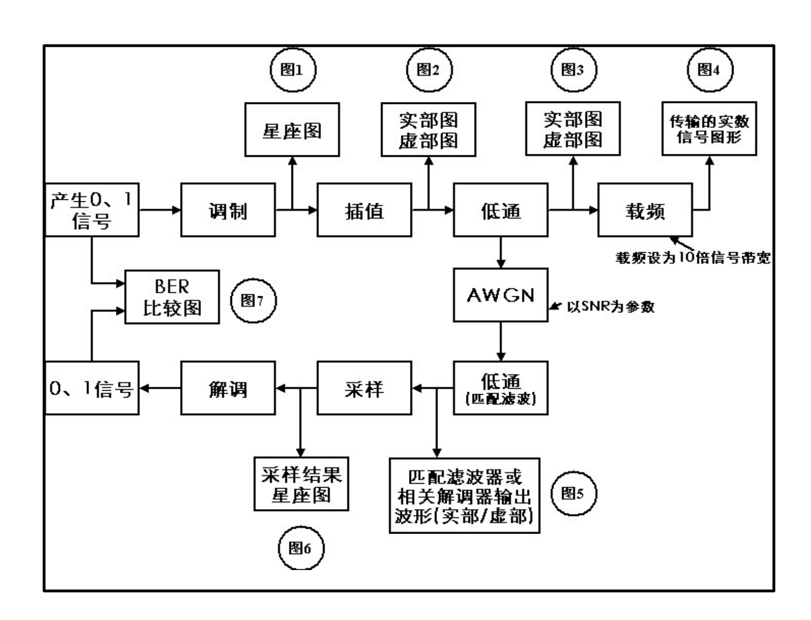


图 4-13　通信系统仿真框图

4.4.1　建立数学模型

建立模型前先对要解决的问题作细致的分析，弄清楚系统的输入/输出和工作过程，然后

逐步建立数学模型。在本例中需要对通信系统中的各个部分进行分析和建模。首先要产生随机数，可以用 MATLAB 自带的函数实现，然后根据格雷码的编码原则实现基带的 QAM 调制。

插值（过采样）是为了后面经过低通时更好的成形滤波的需要，相当于滤波后采样点数增加，从而使波形更平滑。实际的通信系统中有两倍的、四倍的和八倍的过采样方式，本系统中采样八倍过采样，虽然倍数越高得到的波形越平滑，系统的性能越好，但是这样会增加运算量。在设备条件一定的情况下，可能会影响到通信的实时性，尤其是载波调制时，这个问题更为明显，本系统采用 8 倍过采样是一种折中的做法。

实际信道中传输的都是模拟信号，所以插值之后需要进行滤波，实现 D/A 转换，在 MATLAB 中所有的计算都是离散的，这样经过数字滤波器后仍是一个离散序列。程序中采用平方根升余弦滚降低通滤波器来进行发送端的低通滤波和接收端的匹配滤波，滚降系数取 0.5，由于滤波器的响应出现了一个群延迟，所以在作图的时候需要通过将输入信号延迟一下做补偿，补偿值为 *delay*fd*。

信号在信道中传输时会受到噪声的影响，本系统中用叠加高斯白噪声的方法来模拟这个现象，程序中用函数（*awgn*）来实现。

经过信道的传输信号到达接收端，首先要经过匹配滤波器，去掉噪声的影响。对于平方根升余弦滚降低通滤波器来讲，匹配滤波器仍是它本身。

由于过采样的过程中加入了多余的点，而实际发送的信号并不包括这些信号，所以接收的时候应该通过采样的方式将实际发送的信号点提取出来。采样时应该注意与发送信号的同步，这里在滤波的过程中前后分别加了 24 个冗余的点，采样时应该把这些点排除在外。解调部分根据欧氏距离最小的原则对采样后的信号进行判决，然后按照调制时的规则进行逆变换，每一个符号分成两个比特，然后可以计算实际的误比特率 *ber* 等于接收的错误比特数除以总比特数。

对于加载波部分，程序中采用 10 个点表示一个正弦波形。因为要求载波频率是信号频率（指的是 I 路和 Q 路分离后的频率）的 10 倍，随机序列经过 I 路和 Q 路分离后，频率为 1Hz，映射后变为 0.5Hz，插点后频率扩大八倍变为 4Hz，而载波频率为 16Hz，所以两个点之间应该有 4 个周期的三角函数波形，每两个点之间对应有载波的 40 个点。为了做到矩阵运算的匹配需要将信号数组作 40 倍扩展，反映到实际电路中应该加保持电路，而这样会增加很多高频成分，往往加低通滤波对波形进行平滑，本系统中不做考虑。

4.4.2 编写代码

数学模型已经基本讨论完毕，下面介绍实现的代码。整个系统是通过一个主函数和若干个子函数实现的，这样便于调试，也增强了可读性。这些子函数包括 QAM 调制函数（Qam_modulation.m）、绘制星座图（plot_astrology.m）、插值（insert_value.m）、绘制正交信号图（plot_2way.m）、绘制滤波后的信号图（stem_2way.m）、升余弦滤波（rise_cos.m）、调制到载波（modulate_to_high.m）、叠加高斯噪声（generate_noise.m）、采样（pick_sig.m）、解调（demodulate_sig.m）和误码率曲线作图（plot_snr.m）。下面将具体介绍这些函数。

1．主函数：project.m

```
function project(N,p)
```

```
%====N 为待仿真序列的维数
%====p 为产生'1'的概率
%==============================
%首先产生随机二进制序列
source=randsrc(1,N,[1,0;p,1-p]);
%==============================
%对产生的二进制序列进行 QAM 调制
[source1,source2]=Qam_modulation(source);
%===============================
%画出星座图
figure(1);
plot_astrology(source1,source2);
%===============================
%两路信号进行插值
sig_insert1=insert_value(source1,8);
sig_insert2=insert_value(source2,8);
%===============================
%画出两路信号的波形图
figure(2);
plot_2way(sig_insert1,sig_insert2,length(sig_insert1),0.5);
title('两路信号的波形图');
%===============================
%通过低通滤波器
[sig_rcos1,sig_rcos2]=rise_cos(sig_insert1,sig_insert2,0.25,2);
%===============================
%画出两路信号的波形图
figure(3);
plot_2way(sig_rcos1,sig_rcos2,length(sig_rcos1)/4,0.5);
hold on
stem_2way(sig_insert1,sig_insert2,3,0.25,2,length(sig_rcos1)/4);
title('通过低通滤波器后两路信号波形图');
%stem_2way(sig_insert1,sig_insert2,length(sig_insert1)/4,0.5);
%===============================
%====将基带信号调制到高频上
[t,sig_modulate]=modulate_to_high(sig_rcos1,sig_rcos2,0.25,2.5);
figure(4);
plot(t(1:500),sig_modulate(1:500));
%===============================
%====将滤波后的信号加入高斯白噪声
snr=10;
```

```
[x1,x2]=generate_noise(sig_rcos1,sig_rcos2,snr);
sig_noise1=x1';
sig_noise2=x2';
%end;
figure(5)
plot_2way(sig_noise1,sig_noise2,length(sig_noise1)/4,0.5);
%==============================
%====经过匹配滤波器
[sig_match1,sig_match2]=rise_cos(sig_noise1,sig_noise2,0.25,2);
figure(6);
plot_2way(sig_match1,sig_match2,length(sig_match1)/4,0.5);
%==============================
%采样
[x1,x2]=pick_sig(sig_match1,sig_match2,8);
sig_pick1=x1;
sig_pick2=x2;
%画出星座图
figure(7)
plot_astrology(sig_pick1,sig_pick2);
%解调
signal=demodulate_sig(sig_pick1,sig_pick2);
%画出
figure(8)
plot_snr;
```

2．QAM 调制函数 Qam_modulation.m

```
function [y1,y2]=Qam_modulation(x)
%QAM_modulation
%对产生的二进制序列进行 QAM 调制
    %=====首先进行串并转换，将原二进制序列转换成两路信号
N=length(x);
a=1:2:N;
y1=x(a);
y2=x(a+1);
    %=====分别对两路信号进行 QPSK 调制
          %======对两路信号分别进行 2～4 电平变换
a=1:2:N/2;
temp1=y1(a);
temp2=y1(a+1);
y11=temp1*2+temp2;
```

```
temp1=y2(a);
temp2=y2(a+1);
y22=temp1*2+temp2;
            %=======对两路信号分别进行相位调制
a=1:N/4;
y1=(y11*2-1-4)*1.*cos(2*pi*a);
y2=(y22*2-1-4)*1.*cos(2*pi*a);
y1(find(y11==0))=-3;
y1(find(y11==1))=-1;
y1(find(y11==3))=1;
y1(find(y11==2))=3;
y2(find(y22==0))=-3;
y2(find(y22==1))=-1;
y2(find(y22==3))=1;
y2(find(y22==2))=3;
```

3．**绘制星座图** plot_astrology.m

```
function plot_astrology(a,b)
%画出星座图
figure(1)
subplot(1,1,1)
plot(a,b,'*');
axis([-5 5 -5 5]);
line([-5,5],[0,0],'LineWidth',3,'Color','red');
line([0,0],[-5,5],'LineWidth',3,'Color','red');
title('QAM 星座图');
```

4．**插值** insert_value.m

```
function y=insert_value(x,ratio)
%两路信号进行插值
y=zeros(1,ratio*length(x));
a=1:ratio:length(y);
y(a)=x;
```

5．**绘制正交信号图** plot_2way.m

```
function y=plot_2way(x1,x2,len,t)
mm=150;
subplot(2,1,2);
plot((1:len)*t,x2(1:len));
axis([0 len*t -4 4]);
```

```
hold on
plot((1:len)*t,x2(1:len),'.','color','red');
hold off
xlabel('虚部信号');
subplot(2,1,1);
plot((1:len)*t,x1(1:len));
axis([0 len*t -4 4]);
hold on
plot((1:len)*t,x1(1:len),'.','color','red');
xlabel('实部信号');
hold off
```

6．绘制滤波后的信号图 stem_2way.m

```
function stem_2way(x1,x2,delay,fd,fs,len)
subplot(2,1,1);
hold on
stem(((1:len)+fs/fd*3)/fs,x1(1:len));
subplot(2,1,2);
hold on
stem(((1:len)+fs/fd*3)/fs,x2(1:len));
```

7．升余弦滤波 rise_cos.m

```
function [y1,y2]=rise_cos(x1,x2,fd,fs)
[yf, tf]=rcosine(fd,fs, 'fir/sqrt');
[yo1, to1]=rcosflt(x1, fd,fs,'filter/Fs',yf);
[yo2, to2]=rcosflt(x2, fd,fs,'filter/Fs',yf);
y1=yo1;
y2=yo2;
```

8．调制到载波 modulate_to_high.m

```
function [t,y]=modulate_to_high(x1,x2,f,hf)
yo1=zeros(1,length(x1)*hf/f*10);
yo2=zeros(1,length(x2)*hf/f*10);
n=1:length(yo1);
yo1(n)=x1(floor((n-1)/(hf/f*10))+1);
yo2(n)=x1(floor((n-1)/(hf/f*10))+1);
t=(1:length(yo1))/hf*f/10;
y=yo1.*cos(2*pi*hf*t)-yo2.*sin(2*pi*hf*t);
```

9．叠加高斯噪声 generate_noise.m

```
function [y1,y2]=generate_noise(x1,x2,snr)
```

```
snr1=snr+10*log10(4);%符号信噪比
ss=var(x1+i*x2,1);
y=awgn([x1+j*x2],snr1+10*log10(ss/10),'measured');
y1=real(y);
y2=imag(y);
```

10．**采样** pick_sig.m

```
function [y1,y2]=pick_sig(x1,x2,ratio)
y1=x1(ratio*3*2+1:ratio:(length(x1)-ratio*3*2));
y2=x2(ratio*3*2+1:ratio:(length(x1)-ratio*3*2));
```

11．**解调** demodulate_sig.m

```
function y=demodulate_sig(x1,x2)
xx1(find(x1>=2))=3;
xx1(find((x1<2)&(x1>=0)))=1;
xx1(find((x1>=-2)&(x1<0)))=-1;
xx1(find(x1<-2))=-3;
xx2(find(x2>=2))=3;
xx2(find((x2<2)&(x2>=0)))=1;
xx2(find((x2>=-2)&(x2<0)))=-1;
xx2(find(x2<-2))=-3;
temp1=zeros(1,length(xx1)*2);
temp1(find(xx1==-1)*2)=1;
temp1(find(xx1==1)*2-1)=1;
temp1(find(xx1==1)*2)=1;
temp1(find(xx1==3)*2-1)=1;
```

12．**误码率曲线作图** plot_snr.m

```
clear;
%用来仿真 QAM 的误 bit 率
snr=1:1:11;
%先来计算理论误 bit 率
error_theory=(1-(1-(2*(1-1/sqrt(16))*1/2*erfc(1/sqrt(2)*sqrt(3*4*10.^(snr/10)/(16-1))))).^2)/4;
%用理论的误 bit 率来决定需要仿真的点数
N=floor(1./error_theory)*100+100;
N(find(N<5000))=5000;
%开始仿真
global p;
for i=1:length(N);
    %首先产生随机二进制序列
```

```
        source=randsrc(1,N(i),[1,0;p,1-p]);
        %对产生的二进制序列进行 QAM 调制
        [source1,source2]=Qam_modulation(source);
        %插值
        sig_insert1=insert_value(source1,8);
        sig_insert2=insert_value(source2,8);
        [source1,source2]=rise_cos(sig_insert1,sig_insert2,0.25,2);
        %====将滤波后的信号加入高斯白噪声
        [x1,x2]=generate_noise(source1',source2',snr(i));
        sig_noise1=x1';
        sig_noise2=x2';
        [sig_noise1,sig_noise2]=rise_cos(sig_noise1,sig_noise2,0.25,2);
        [x1,x2]=pick_sig(sig_noise1,sig_noise2,8);
        sig_noise1=x1;
        sig_noise2=x2;
        %解调
        signal=demodulate_sig(sig_noise1,sig_noise2);
        %计算误 bit 率
        error_bit(i)=length(find(signal-source)~=0)/N(i);
    end;
    %画出图形
    semilogy(snr,error_bit,'-b');
    hold on
    semilogy(snr,error_theory,'-r')
```

4.4.3 运行程序

运行程序，即在命令行中输入 project(40000,0.5)，此数值可以根据需要进行选择。如果有错误则进行调试。正确的运行结果如图 4-14～图 4-21 所示。

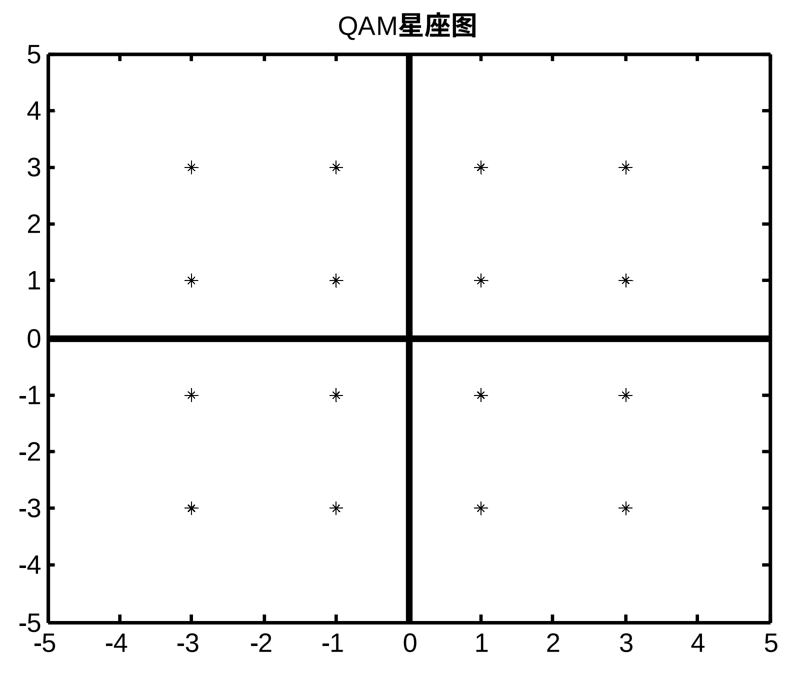


图 4-14　QAM 星座图

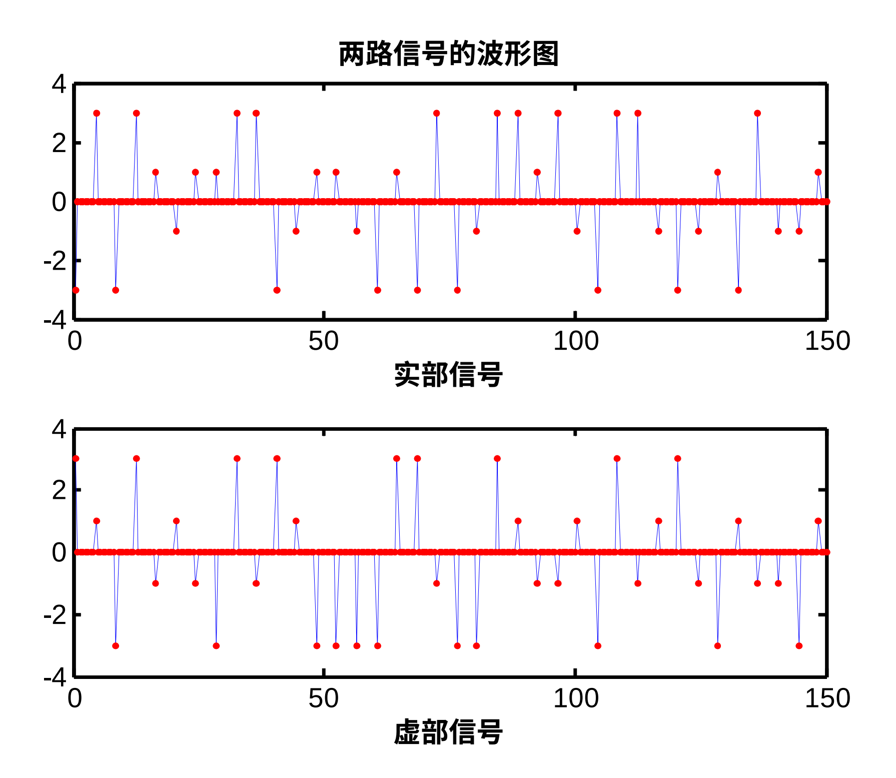


图 4-15　经过插值后的两路信号波形图

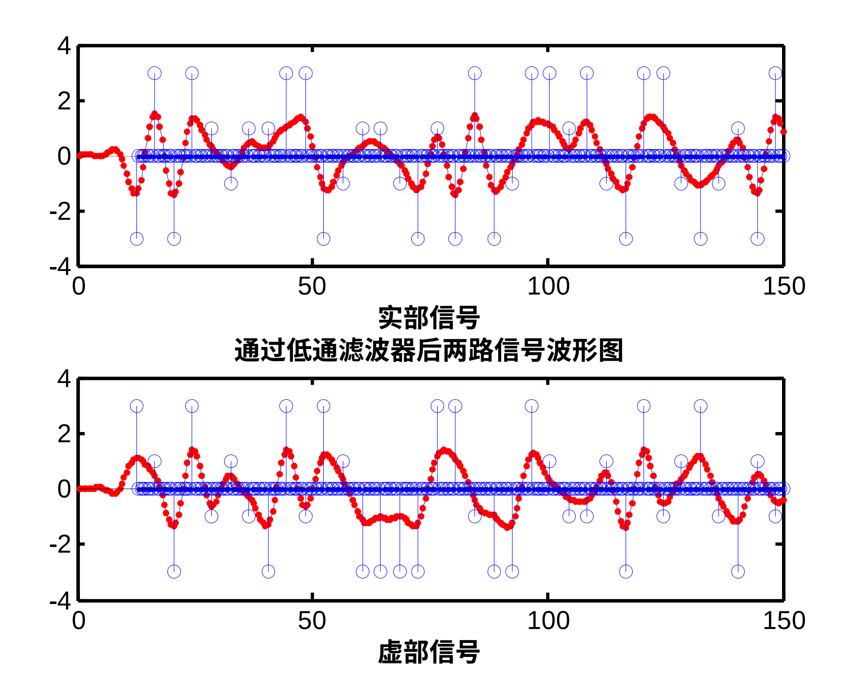


图 4-16　通过低通滤波后的两路信号波形图

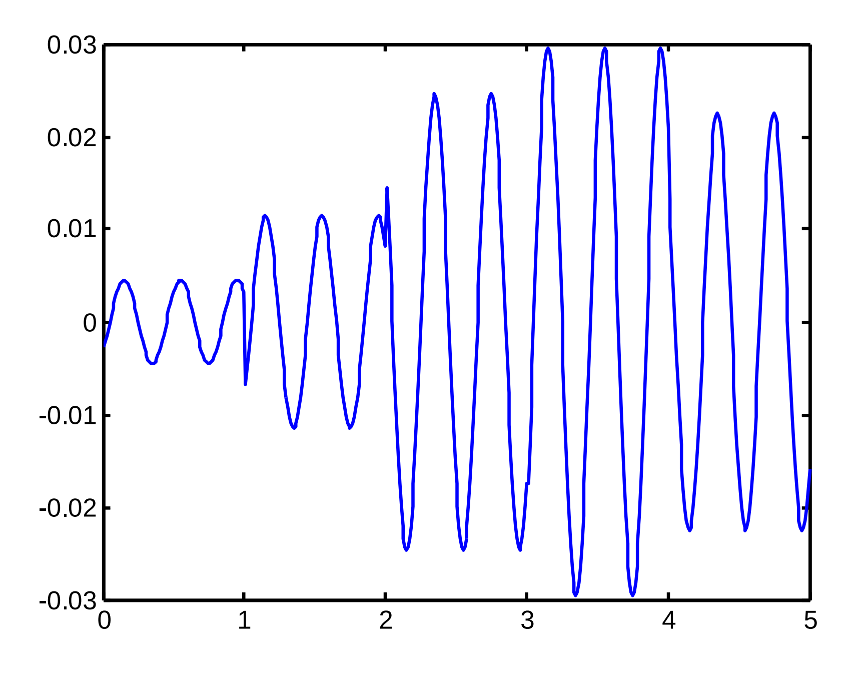


图 4-17　载波调制信号图

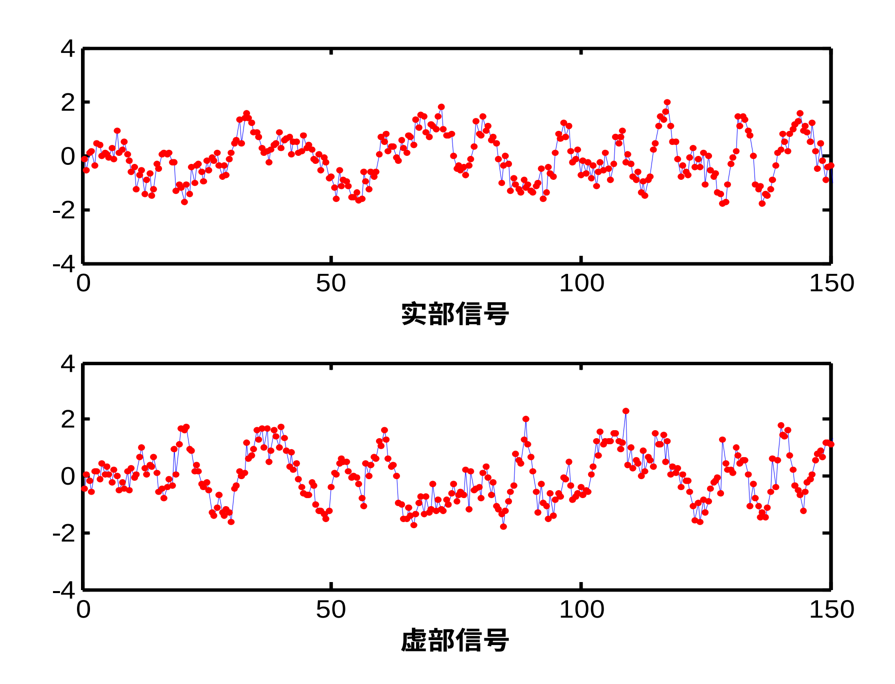


图 4-18　加入高斯白噪声的两路信号波形

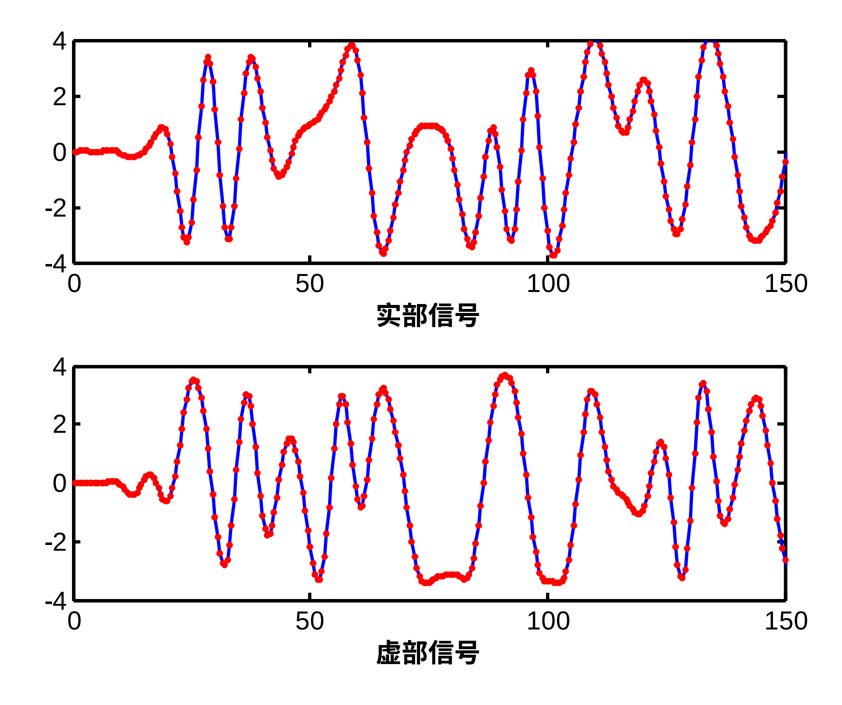


图 4-19　经过匹配滤波器后的波形

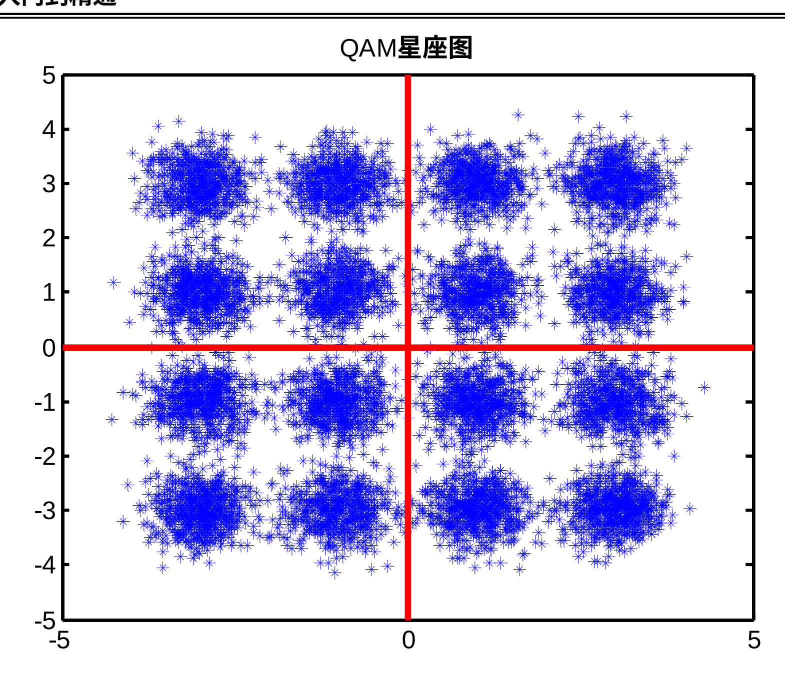


图 4-20　信噪比为 10dB 时的星座图

由图 4-20 可见，当信噪比为 10dB 时，解调出来的信号大部分是正确的，基本上集中在发送信号的周围。

图 4-21 中的横坐标是信噪比，信噪比越大，误码率越低，这一点符合通信系统的规律。（其中上面一条是实际的误码率曲线，下面一条是理论曲线，二者基本吻合，说明程序是正确的。）

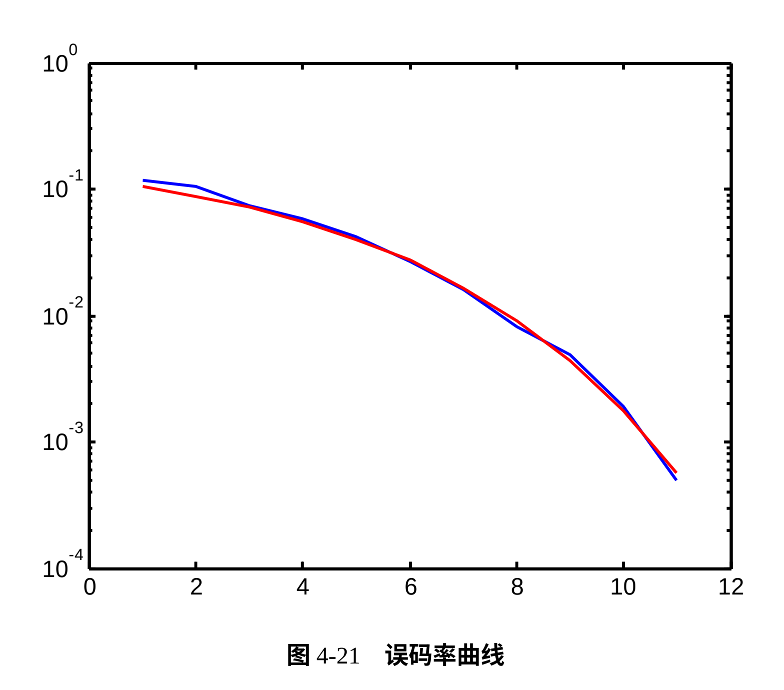


图 4-21　误码率曲线

从本例中可以看出，MATLAB 系统仿真的功能是十分强大的，重要的是建立正确的数学模型。在编写程序时，应尽量将重复用到的部分编写成独立的函数。同时，为了程序的可扩展性，应该尽量避免使用全局变量。各个函数可单独调试，调试正确后再统一运行，如果仍不正确，则需要认真考虑算法是否合理。

第 5 章　数据可视化

不管根据计算得到的数据堆或是符号堆是多么的准确，人们还是很难从这一大堆原始的数据和符号中发现它们的具体物理含义或是内在规律，而数据图形恰能使视觉感官直接感受到数据的许多内在本质，发现数据的内在联系。因此数据可视化是一项非常重要的技术。

MATLAB 可以表达出数据的二维、三维和四维的图形。通过对图形的线型、立面、色彩、光线、视角等属性的控制，可把数据的内在特征表现得更加细腻完善。下面将向读者介绍这些图形绘制和图形处理的命令。

5.1　二维绘图

二维图形的绘制是 MATLAB 语言图形处理的基础，也是在绝大多数数值计算中广泛应用的图形方式之一。本小节主要向读者介绍 plot、fplot、ezplot 三个基本的二维绘图命令。

5.1.1　plot 命令

绘制二维图形最常用的函数就是 plot 函数，通过不同形式的输入，该函数可以实现不同的功能。其调用格式有如下 3 种：

1．plot(*y*)

此命令中参数 *y* 可以是向量、实数矩阵或复数向量。若 *y* 为向量，则绘制的图形以向量索引为横坐标值、以向量元素的值为纵坐标值；若 *y* 为实数矩阵，则绘制 *y* 的列向量对其坐标索引的图形；若 *y* 为复向量，则 plot(*y*)相当于 plot(real(*y*)，imag(*y*)。在后面介绍的两种调用格式中，元素的虚部将被忽略。

例如用 plot(*y*)命令绘制向量，如图 5-1 所示，具体代码设置如下：

```
>>t=1:0.1:10;
>>y=sin(t);
>>plot(y)
```

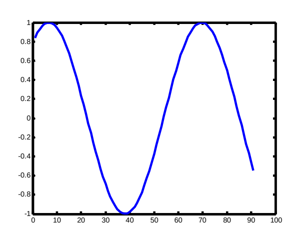


图 5-1　plot(*y*)绘制向量

注意：图 5-1 中横坐标的范围不是[1，10]，而是[1，91]，因为 plot(*y*)以向量索引为横坐标值。

例如用 plot(*y*)命令绘制矩阵，如图 5-2 所示，具体代码设置如下：

```
>>y= [0 1 2;2 3 4;5 6 7]
>>plot(y)
```

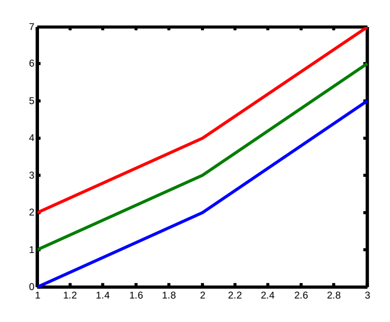


图 5-2　plot（y）绘制矩阵

例如用 plot(*y*)命令绘制复向量，如图 5-3 所示，具体代码设置如下：

```
>>x=[1:1:100];
>>y=[2:2:200];
>>z=x+y.*i;
>>plot(z)
```

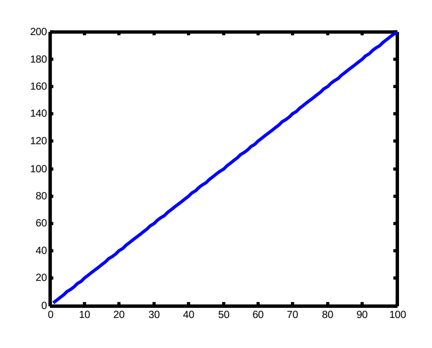


图 5-3　plot（y）绘制复向量

2．plot(***x***, ***y***)

x，***y*** 均可为向量和矩阵，其中有 3 种组合用于绘制连线图。

x，***y*** 均为 *n* 维向量时，绘制向量 ***y*** 对向量 ***x*** 的图形，即以 ***x*** 为横坐标，***y*** 为纵坐标。

x 为 *n* 维向量，***y*** 为 *m*×*n* 或 *n*×*m* 的矩阵时，该命令将在同一图内绘得 *m* 条不同颜色的连线。图中以向量 *x* 为 *m* 条连线的公共横坐标，纵坐标为 ***y*** 矩阵的 *m* 个 *n* 维分量。

x，***y*** 均为 *m*×*n* 矩阵时，将绘得 *n* 条不同颜色的连线。绘制规则为：以 ***x*** 矩阵的第 *i* 列分量作为横坐标，矩阵 ***y*** 的第 *i* 列分量作为纵坐标，绘得第 *i* 条连线。

例如用 plot(*x*, *y*)绘制双向量，如图 5-4 所示，代码设置如下：

```
>>x=0:0.1:10;
>>y=sin(x)+2;
```

```
>>plot(x,y)
```

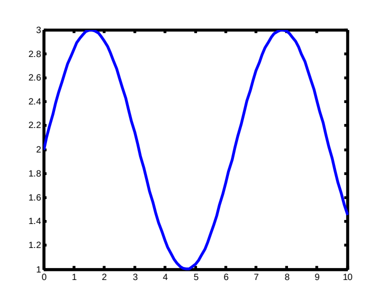


图 5-4　plot(*x* ,*y*)绘制双向量

例如用 plot(*x*，*y*)绘制向量和矩阵，如图 5-5 所示，代码设置如下：

```
>>x=0:0.1:10;
>>y=[sin(x)+2;cos(x)+1];
>>plot(x,y)
```

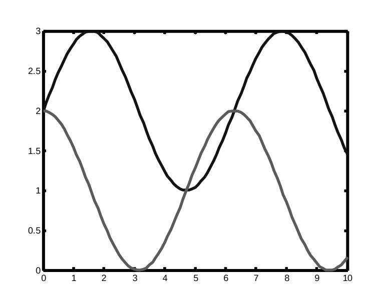


图 5-5　plot(*x* ,*y*)绘制向量和矩阵

例如用 plot(*x*, *y*)绘制双矩阵，如图 5-6 所示，代码设置如下：

```
>>x=[1 2 3;4 5 6;7 8 9;2 3 4;5 6 7];
>>y=[2 4 5;3 6 7;4 6 8;1 3 5;2 6 3];
>>plot(x,y)
```

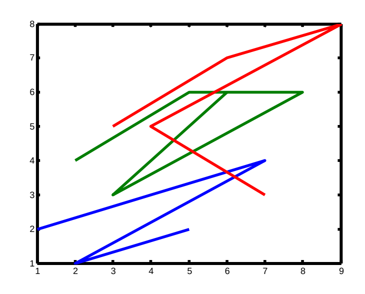


图 5-6　plot(*x*, *y*)绘制双矩阵

3．plot(***x***, ***y***, ***s***)

此格式用于绘制不同的线型、点标和颜色的图形，其中 *s* 为字符，可以代表不同的线型、点标和颜色。常见的可用字符及其意义如表 5-1 所示。

表 5-1　　　　　　　　　　　二维绘图的图形常用设置选项

选项	说明	选项	说明
-	实线	.	点
:	点线	o	圆
-.	点划线	+	加号
--	虚线	*	星号
y	黄色	x	x 符号
m	紫红色	s	方形
c	蓝绿色	d	菱形
r	红色	v	下三角
g	绿色	^	上三角
b	蓝色	<	左三角
w	白色	>	右三角
k	黑色	p	正五边形

例如用 plot(*x*, *y*, *s*)绘图，如图 5-4 所示，代码设置如下：

```
>>x=0:0.5:20;
>>y=sin(x);
>>plot(x,y,’-.rd’)
```

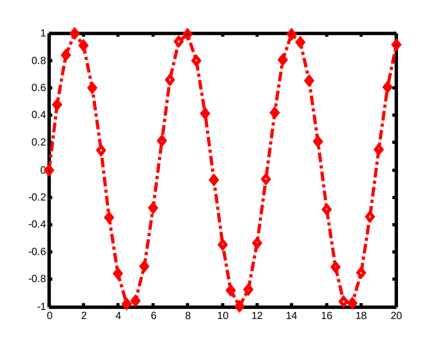


图 5-7　plot（*x*, *y*, *s*）绘图

5.1.2　fplot 命令

前面介绍的 plot 命令是将从外部输入或通过函数数值计算得到的数据矩阵转化为连线图。而在实际的应用中，用户可能并不知道某一个函数随自变量变化的趋势，此时若采用 plot 命令来绘图，则有可能会因为自变量的取值间隔不合理而使曲线图形不能反应出自变量在某些区域内函数值的变化情况。当然用户可以将自变量间隔取得足够小以体现函数值随自变量变化的精确曲线，但是这样会使数据量变大。

而 fplot 命令就可以很好地解决这个问题。fplot 命令用于指导如何通过函数来取得绘图的数值点矩阵。该命令通过内部的自适应算法来动态决定自变量的取值间隔，当函数值变化缓慢时，间隔取大一点；变化剧烈时（即函数的二阶导数很大），间隔取小一点。

该命令的使用格式如下：

```
fplot(function,limits)
fplot(function,limits,LineSpec)
fplot(function,limits,tol)
fplot(function,limits,tol,LineSpec)
fplot(function,limits,n)
fplot(axes_handle,...)
[X,Y] = fplot(function,limits,...)
[...] = fplot(function,limits,tol,n,LineSpec,P1,P2,...)
```

下面介绍其中各项参数的含义。

- function：待绘制的函数的名称；
- limits：定义 *x* 轴（自变量）的取值范围，或 *x* 轴和 *y* 轴（应变量）的范围[xmin xmax]，[xmin xmax ymin ymax]；
- LineSpec：定义绘图的线型、颜色和数据点等（如表 5-1 所示）；
- tol：相对误差容忍度，默认值为 2e-3；
- n：当 n≥1 时，至少绘制 n+1 个点，默认值为 1。最大的步长限制为(1/n)*(xmax-xmin)；
- axes_handle：坐标轴句柄，函数的图形将绘制在这个坐标系中；
- P1,P2…：向函数传递参数值。

【例如】

- 创建函数，并保存为 myfun.m，具体代码设置如下：

```
function Y= myfun(x)
Y=200*sin(1./tan(pi.*x));
```

- 用 fplot 命令求得坐标点，并按同样大小创建一个等间隔坐标点，代码设置如下：

```
>>[X,Y]=fplot('myfun',[-0.1 0.1],2e-4)
>>L=size(X);
>>m=-0.1:0.2/(L(1)+1):0.1;
>>n=myfun(m);
```

- 作图比较 fplot 命令与 plot 命令，代码设置如下：：

```
>>plot(X,Y)
>>figure    //重新打开一个绘图窗口
>>plot(m,n)
```

用 fplot 命令和 plot 命令绘制出的结果如图 5-8 和图 5-9 所示。

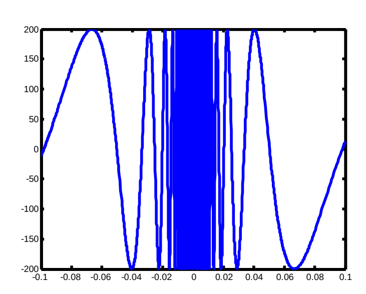


图 5-8　fplot 命令绘图结果

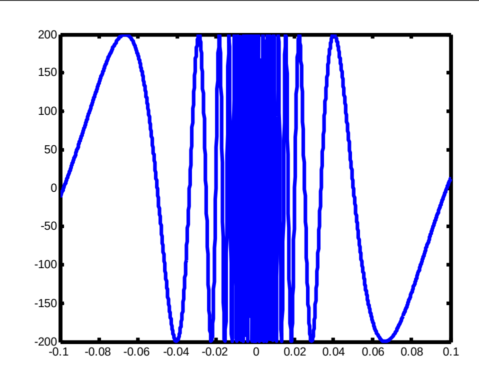


图 5-9　plot 命令绘图结果

5.1.3　ezplot 命令

ezplot 命令也是用于绘制函数在某一自变量区域内的图形。该命令的使用格式如下：

```
ezplot(f)
ezplot(f,[min,max])
ezplot(f,[xmin,xmax,ymin,ymax])
ezplot(x,y)
ezplot(x,y,[tmin,tmax])
ezplot(...,figure_handle)
ezplot(axes_handle,...)
h = ezplot(...)
```

当 f = f(x)时，各参数的含义如下。

- ezplot(f)：绘制表达式 f=f(x)在默认区域 2*pi<x<2.*pi 内的图形，f 为函数句柄；
- ezplot(f,[min,max])：绘制表达式 f=f(x)在 min<x<max.区域内的图形；

当 f=f(x,y)时，各参数的含义如下。

- ezplot(f)：绘制函数 f(x,y)=0 在默认区域 2*pi<x<2.*pi，2*pi<y<2.*pi 内的图形；
- ezplot(f,[xmin,xmax,ymin,ymax])：绘制 f(x,y) = 0 在区域 xmin<x<xmax、ymin<y<ymax 内的图形；
- ezplot(f,[min,max])：绘制 f(x,y) = 0 在区域 min<x<maxa、min<y<max 内的图形；

> 注意：如果函数或表达式的变量名不是 x,y，如 zplot('u^2-v^2-1',[-3,2,-2,3])就是绘制 u^2-v^2-1=0 在-3<u<2,-2<v<3 区域内的图形。

- ezplot(x,y)：绘制参数方程组 x=x(t)，y =y(t)在默认区域 0<t<2.*pi 内的图形；
- ezplot(x,y,[tmin,tmax])：绘制 x =x(t)，y=y(t)在区域 tmin<t<tmax 内的图形；
- ezplot(...,figure_handle)：在句柄为 figure_handle 的窗口中绘制给定函数在给定区域内的图形；
- ezplot(axes_handle,...)：在句柄为 axes_handle 的坐标系上绘制图形；
- h = ezplot(...)：返回直线对象的句柄到 *h* 变量中。

【例如】

绘制非显式函数 $x^2+y^2-4=0$ 在区域[-3,3,-3,3]内的图形。

```
>>ezplot(' x^2+y^2-4', [-3,3,-3,3]);
```

用 ezplot 命令绘制出来的结果如图 5-10 所示。

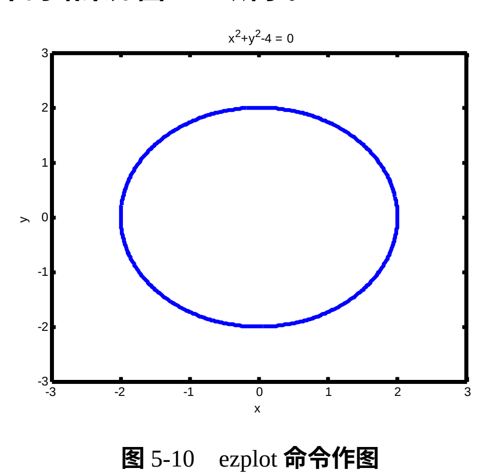


图 5-10　ezplot 命令作图

5.2　三维绘图

在实际的工程计算中常常需要将结果表示成三维图形，MATLAB 语言为此提供了相应的三维图形的绘制功能。这些绘制功能与二维图形的绘制有很多类似之处，其中曲线的属性设置完全相同。最常用的三维绘图是绘制三维曲线图、三维网格图和三维曲面图 3 种基本类型，相应的 MATLAB 命令为 plot3、mesh 和 surf，下面分别介绍它们的具体使用方法。

5.2.1　plot3 命令

与 plot 类似，plot3 是三维绘图的基本函数，其调用格式如下：

```
plot3(X1,Y1,Z1,...)
plot3(X1,Y1,Z1,LineSpec,...)
plot3(...,'PropertyName',PropertyValue,...)
h=plot3(...)
```

其中 X1,Y1,Z1 为向量或矩阵，LineSpec 定义曲线线型、颜色和数据点等（参见表 5-1），PropertyName 为线对象的属性名，PropertyValue 为相应属性的值，*h* 是用于存放曲线簇中每一个线对象的句柄的变量。

当 X1,Y1,Z1 为长度相同的向量时，plot3 命令将绘得一条分别以向量 X1,Y1,Z1 为 *x*，*y*，*z* 轴坐标值的空间曲线；

当 X1,Y1,Z1 均为 $m\times n$ 的矩阵时，plot3 命令将绘得 *m* 条曲线。其第 *i* 条空间曲线分别以 X1,Y1,Z1 矩阵的第 *i* 列分量为 *x*，*y*，*z* 轴坐标值的空间曲线。

例如用 plot3 命令绘制螺旋线，如图 5-11 所示，代码设置如下：

```
>>t=0:pi/50:10*pi;
>>plot3(cos(t),sin(t),t)
```

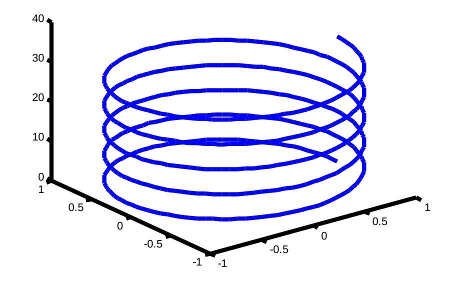


图 5-11　螺旋线

例如用 plot3 命令绘制向量，如图 5-12 所示，具体代码如下：

```
>>t=[0:pi/100:2*pi];
>>x=[sin(t) sin(t)];
>>y=[cos(t) cos(t)];
>>z=[(sin(t)).^2+(cos(t)).^2    (sin(t)).^2+(cos(t)).^2+1];
>>plot3(x,y,z)
```

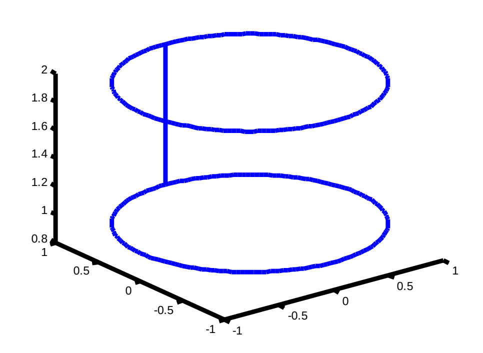


图 5-12　plot3 绘制向量

5.2.2　mesh 命令

该命令与 plot3 不同的是它可以绘出在某一区间内完整的曲面，而不是单根曲线。其调用格式如下：

```
mesh(X,Y,Z)
mesh(Z)
mesh(...,C)
mesh(...,'PropertyName',PropertyValue,...)
mesh(axes_handles,...)
h = mesh(...)
hsurface = mesh('v6'...)
```

其中 C 用于定义颜色，如果没有定义 C，则 mesh(X,Y,Z)绘制的颜色随 Z 值（即曲面高度）成比例变化。X 和 Y 必须均为向量，若 X 和 Y 的长度分别为 m 和 n，则 Z 必须为 $m \times n$ 的矩阵，也即[m,n]=size(Z)，在这种情况下，网格线的顶点为（X(j)，Y(i)，Z(i,j)）；若参数中

没有提供 *X*，*Y*，则将(i,j)作为 Z（i，j）的 *X*，*Y* 轴坐标值。

例如用 mesh 函数绘制三维曲线，如图 5-13 所示，代码设置如下：

```
>>x=-4:0.1:4;y=x';
>>m=ones(size(y))*x;
>>n=y* ones(size(x))
>>p=sqrt(m.^2+n.^2)+eps;
>>z=sin(p)./p
>>mesh(z)
```

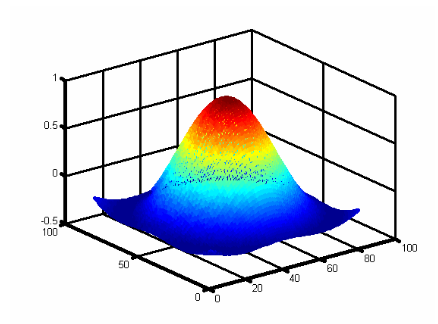


图 5-13　mesh 函数绘制三维曲面

例如用 mesh(z)函数绘制三维曲面，如图 5-14 所示，代码设置如下：

```
>>x=[0:0.1:5;2:0.1:7];
>>mesh(x)
```

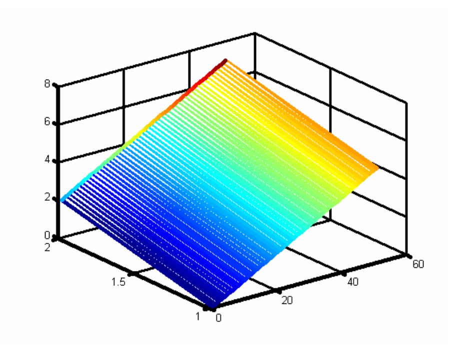


图 5-14　mesh（*z*）函数绘制三维曲面

5.2.3　surf 命令

该命令的调用方法与 mesh 命令类似，不同的是 mesh 函数绘制的图形是一个网格图，而 surf 命令绘制得到的是着色的三维曲面。着色的方法是在得到相应的网格后，对每一个网格依据该网格所代表的节点的色值（由变量 C 控制）来定义这一网格的颜色。调用方式如下：

```
surf(Z)
surf(X,Y,Z)
surf(X,Y,Z,C)
surf(...,'PropertyName',PropertyValue)
```

```
surf(axes_handle,...)
h = surf(...)
hsurface = surf('v6',...)
```

其中各个参数意义与 mesh 命令中的相同。

例如分别用 surf 命令和 mesh 命令绘制着色曲面图进行比较，如图 5-15 和图 5-16 所示。代码设置如下：

```
>>[X,Y,Z] = peaks(30);
>>surf(X,Y,Z)
>> figure
>> mesh(Z)
```

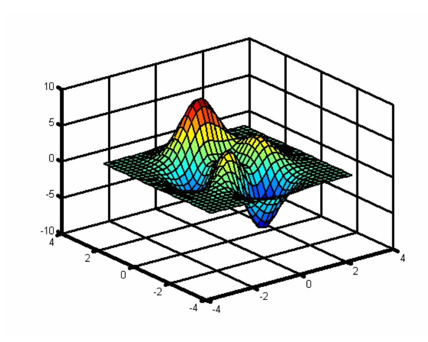


图 5-15　用 surf 命令绘制着色曲面图

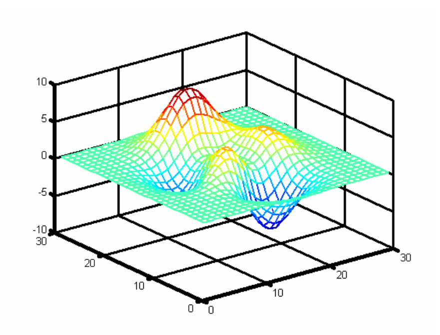


图 5-16　mesh 命令绘制相同的数据

5.2.4　基本三维绘图命令的改进命令

前面介绍了 3 个基本的三维绘图命令，下面向读者介绍另外几个常用命令，它们对基本的三维绘图命令增加了一些特别的处理图形的功能。

1．meshgrid、meshc 和 meshz

meshgrid 命令的作用是将给定的区域按一定的方式划分成平面网格，该平面网格可以用来绘制三维曲面，调用方法如下：

```
[X,Y]=meshgrid(x,y)
```

其中 x 和 y 为给定的向量，是用来定义网格划分区域的，也可定义网格划分方法；X,Y 用来存储网格划分后的数据矩阵。

meshc 是在用 mesh 命令绘制的三维曲面图下绘出等高线，调用方式与 mesh 相同。

例如用 meshc 命令绘制三维曲面，如图 5-17 所示。代码设置如下：

```
>>[X,Y,Z] = peaks(30);
>> meshc(Z)
```

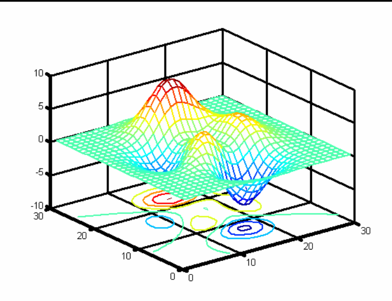


图 5-17　meshc 命令绘制三维曲面

meshz 是在 mesh 函数的作用之上增加绘制边界面的功能，调用方式与 mesh 一样。例如用 meshz 命令绘制三维曲面，如图 5-18 所示。代码设置如下：

```
>>[X,Y,Z] = peaks(30);
>> meshz(Z)
```

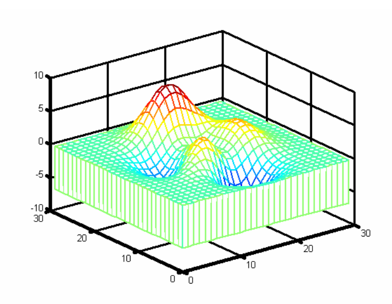


图 5-18　meshz 命令绘制三维曲面

2．surfc

surfc 与 meshc 类似，在 surf 命令绘制的图形上加绘等高线，调用方式与 surf 命令一样。例如用 surfc 命令绘制三维曲面，如图 5-19 所示。代码设置如下：

```
>>[X,Y,Z] = peaks(30);
>> surfc(Z)
```

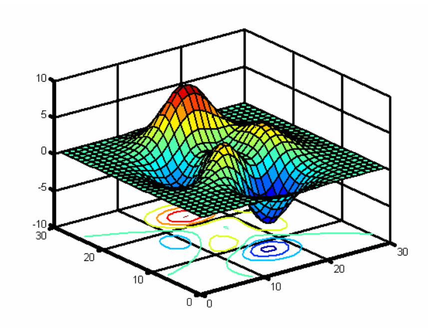


图 5-19　surfc 命令绘制三维曲面

3．waterfall

该命令用于绘制形似瀑布流水形状的网线图，调用格式与 mesh 相同。

例如用 waterfall 命令绘制三维曲面，如图 5-20 所示。代码设置如下：

```
>>[X,Y,Z] = peaks(30);
>> waterfall(Z)
```

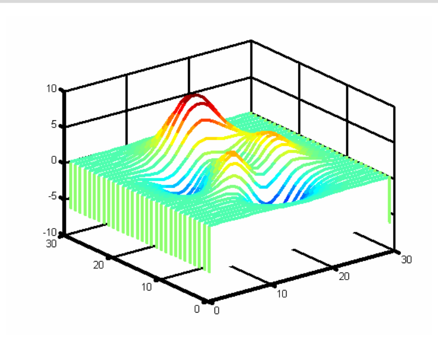


图 5-20　waterfall 命令绘制三维曲面

5.3　特殊图形

5.3.1　二维特殊图形函数

常见的特殊二维图形函数如表 5-2 所示。

表 5-2　二维特殊图形函数

函数名	说明	函数名	说明
area	填充绘图	fplot	函数绘制
bar	条形图	hist	柱状图
barh	水平条形图	pareto	Pareto 图
comet	彗星图	pie	饼状图
errorbar	误差带图	plotmatrix	分散矩阵绘制
ezplot	简单绘制函数图	ribbon	三维图的二维条状显示
ezpolar	简单绘制极坐标图	scatter	散射图
feather	矢量图	stem	离散序列火柴杆状图
fill	多边形填充	stairs	阶梯图
gplot	拓扑图	rose	极坐标系下的柱状图
compass	与 feather 功能类似的矢量图	quiver	向量场

表中的函数均有不同的调用方法，下面通过几个示例向读者介绍其中几个常用的命令，其他函数的使用方法读者可以查阅 MATLAB 7.0 的在线帮助。

【bar 命令】

该命令用于绘制二维垂直条形图，用垂直条形显示向量或矩阵中的值。调用格式如下：

```
bar(y)　//为每一个 y 中的元素画一个条状
bar(x,y)　//在指定的横坐标 x 上画出 y，其中 x 为严格单增的向量。若 y 为矩阵，则 bar 把矩阵分解成
几个行向量，在指定的横坐标处分别画出。
```

bar(...,width)　//设置条形的相对宽度和控制在一组内条形的间距。默认值为 0.8，所以如果用户没有指定 x，则同一组内的条形有很小的间距，若设置 width 为 1，则同一组内的条形相互接触

bar(...,'style')　//style 定义条的形状类型，可以取值'group'或'stack'。其中"group"为默认的显示模式。"group"表示若 *y* 为 n*m 阶的矩阵，则 bar 显示 n 组，每组有 m 个垂直条形的条形图。"stack"表示对矩阵 *y* 的每一个行向量显示在一个条形中，条形的高度为该行向量中的分量和。其中同一条形中的每个分量用不同的颜色显示出来，从而可以显示每个分量在向量中的分布

bar(...,'bar_color')　//'bar_color'定义条的颜色

bar(axes_handle,...)

h = bar(…)　//返回一个 patch 图形对象句柄的向量。每一条形对应一个句柄

【示例 1】

```
>>x=-2.9:0.2:2.9;
>>bar(x,exp(-x.*x),'r')
```

用函数 bar 绘制的条形图如图 5-21 所示。

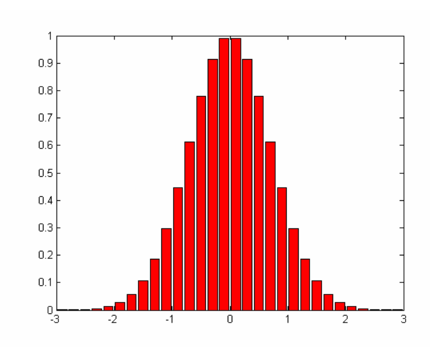


图 5-21　条形图

【示例 2】

```
>>Y = round(rand(5,3)*10);
>>subplot(2,2,1)
>>bar(Y,'group')
>>title 'Group'
>>subplot(2,2,2)
>>bar(Y,'stack')
>>title 'Stack'
>>subplot(2,2,3)
>>barh(Y,'stack')
>>title 'Stack'
>>subplot(2,2,4)
>>bar(Y,1.5)
>>title 'Width = 1.5'
```

函数 bar 绘图命令如图 5-22 所示。

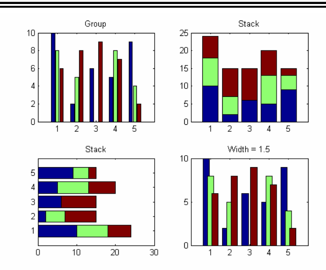


图 5-22　bar 绘图命令

【pie 命令】

该命令用于绘制饼形图。调用格式如下：

```
pie(x)
pie(x, explode)   // explode 与 x 同维的矩阵，若其中有非零元素，x 矩阵中的相应位置的元素在饼图中对应的扇形将向外移出一些，加以突出
pie(...,labels)   // labels 用于定义相应块的标签
pie(axes_handle,...)
h = pie(...)
```

【示例 1】

```
>>x = [1 3 0.5 2.5 2];
>>explode = [0 1 0 0 0];
>>pie(x,explode)
```

用 pie 命令绘制饼状图如图 5-23 所示。

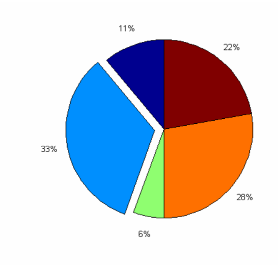


图 5-23　pie 绘制饼状图

【示例 2】

```
>>pie(1:3,{'Taxes','Expenses','Profit'})
```

函数 pie 绘制命令如图 5-24 所示。

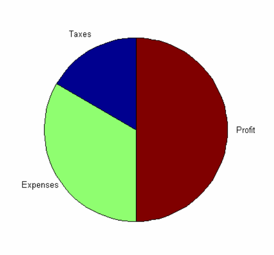


图 5-24　pie 绘图命令

【hist 命令】

该命令用于绘制二维条形直方图，可以显示出数据的分布情况。所有向量 *y* 中的元素或者是矩阵 *y* 的列向量中的元素是根据它们的数值范围来分组的，每一组作为一个条形进行显示。条形直方图中的 *x* 轴反映了数据 *y* 中元素数值的范围，直方图的 *y* 轴显示出参量 *y* 中的元素落入该组的数目。调用格式如下：

n = hist(*y*)　//把向量 *y* 中的元素放入等距的 10 个条形中，且返回每一个条形中的元素个数。若 *y* 为矩阵，则该命令按列对 *y* 进行处理

n = hist(*y*,*x*)　//参量 x 为向量，把 *y* 中元素放到 m（m=length(x)）个由 *x* 中元素指定的位置为中心的条形中

n = hist(*y*,*nbins*)　//参量 *nbins* 为标量，用于指定条形的数目

[n,xout] = hist(...)　//返回向量 *n* 与包含频率计数与条形的位置向量 xout，用户可以用命令 bar(xout,n)画出条形直方图

hist(...)

hist(axes_handle,...)

【示例】

```
>>x = -2.9:0.1:2.9;
>>y = randn(10000,1);
>>hist(y,x)
```

用 hist 命令绘制图形如图 5-25 所示。

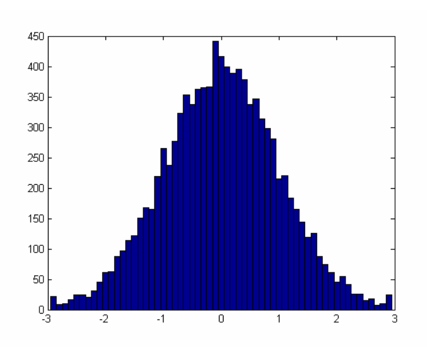


图 5-25　hist 命令绘图

【contour 命令】

该命令用于绘制等高线图。调用格式如下：

```
contour(Z)
contour(Z,n)
contour(Z,v)
contour(X,Y,Z)
contour(X,Y,Z,n)
contour(X,Y,Z,v)
contour(...,LineSpec)
[C,h] = contour(...)
[C,h] = contour('v6',...)
```

其中，输入变量 Z 必须为一数值矩阵，是必须输入的变量；*n* 为所绘图形等高线的条数；*v* 为向量，等高线条数等于该向量的长度，并且等高线的值为对应向量的元素值；*c* 为等高线矩阵；*h* 为等高线的句柄。若没有选择，系统将自动为矩阵 *z* 绘制等高线图。类似的函数有 contourf，用以绘制填充的等高线图，调用格式与 contour 相同。

【示例】

```
>>[X,Y] = meshgrid(-2:.2:2,-2:.2:3);
>>Z = X.*exp(-X.^2-Y.^2);
>>[C,h] = contour(X,Y,Z);
>>set(h,'ShowText','on','TextStep',get(h,'LevelStep')*2)
```

用 contour 命令绘制等高线如图 5-26 所示。

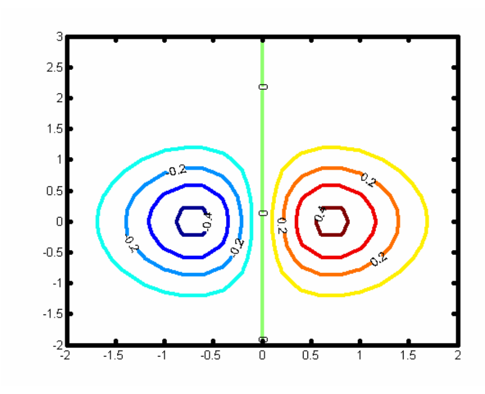


图 5-26　contour 命令绘制等高线

【quiver 命令】

该命令用于绘制矢量图或速度图，绘制向量场的形状。调用格式如下：

quiver(x,y,u,v)　//在坐标（*x*,*y*）点处用箭头图形绘制向量，（*u*，*v*）为相应点的速度分量。*X*，*y*，*u*，*v* 必须具有相同的大小

quiver(u,v)

quiver(...,scale)　// scale 是用来控制看到的图中向量“长度”的实数，默认值为 1。有时需要设置小的值，以免绘制的向量彼此重叠掩盖

quiver(...,LineSpec)

quiver(...,LineSpec,'filled')

quiver(axes_handle,...)

h = quiver(...)

hlines = quiver('v6',...)

【示例】

```
>>[X,Y] = meshgrid(-2:.2:2);
>>Z = X.*exp(-X.^2 - Y.^2);
>>[DX,DY] = gradient(Z,.2,.2);
>>contour(X,Y,Z)
>>hold on
>>quiver(X,Y,DX,DY)
```

用 quiver 绘制矢量图如图 5-27 所示。

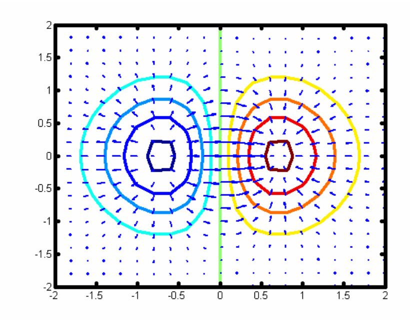


图 5-27　quiver 命令绘制矢量图

【comet 命令】

该命令用于绘制二维彗星图。彗星图为彗星头（一个小圆圈）沿着数据点前进的动画，彗星体为跟在彗星头后面的痕迹，轨道为整个函数的实线。要注意的是，由命令 comet 生成的轨迹图的擦除模式（EraseMode）属性值为 none，该属性使得用户不能打印该图形（只能得到彗星头），且当用户改变窗口的大小时，动画将消失。调用格式如下：

```
comet(y)    //彗星图动画显示向量 y 确定的路线
comet(x,y)    //彗星图动画显示向量 x 与 y 确定的路线
comet(x,y,p)    //指定彗星体的长度 p*length(y)，默认的 p 值为 0.1
comet(axes_handle,...)
```

【示例】

```
>>t = 0:.01:2*pi;
>>x = cos(2*t).*(cos(t).^2);
>>y = sin(2*t).*(sin(t).^2);
>>comet(x,y);
```

用 comet 命令绘制彗星图如图 5-28 所示。

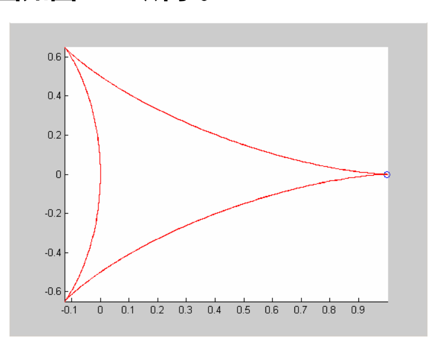


图 5-28　comet 命令绘制彗星图

【errorbar 命令】

该命令用于绘制沿着一曲线画误差棒形图。误差棒为数据的置信水平或者为沿着曲线的偏差。命令输入参数为矩阵时，则按列画出误差棒。调用格式如下：

```
errorbar(Y,E)    //画出向量 y，同时显示在向量 y 的每一元素之上的误差棒。误差棒为 E(i)在曲线 y 上面与下面的距离，所以误差棒的长度为 2*E(i)
```

```
errorbar(X,Y,E)    // X,Y,E 必须为同型参量。若同为向量，则画出带长度为 2*E(i)、对称误差棒于曲线点(X(i),Y(i))之处；若同为矩阵，则画出带长度为 E(i，j)、对称误差棒于曲面点(X(i,j)，Y(i,j))之处
errorbar(X,Y,L,U)    // X,Y,L,U 必须为同型参量。若同为向量，则在点(X(i),Y(i))处画出向下长为 L(i)，向上长为 U(i)的误差棒；若同为矩阵，则在点(X(i,j)，Y(i.j))处画出向下长为 L(i,j),向上长为 U(i,j)的误差棒
errorbar(...,LineSpec)    //用 LineSpec 指定的线型、标记符和颜色等画出误差棒
h = errorbar(...)    //返回线图形对象的句柄向量给 h
```

【示例】

```
>>x=[0:0.2:4*pi];
>>y=sin(x);
>>e=[0:1/(length(x)-1):1];
>>errorbar(x,y,e)
```

用 errorbar 命令绘制误差棒形图如图 5-29 所示。

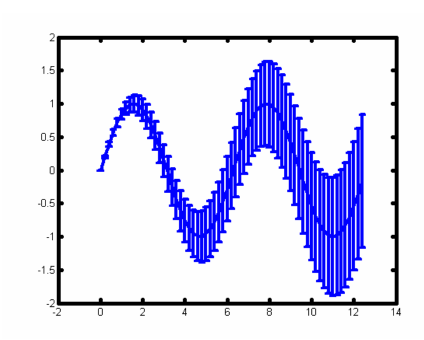


图 5-29　errorbar 命令绘制误差棒形图

5.3.2　特殊的三维图形函数

特殊的三维图形函数如表 5-3 所示。

表 5-3　三维特殊图形函数

函数名	说明	函数名	说明
bar3	三维条形图	surfc	着色图与等高线图结合
comet3	三维彗星轨迹图	trisurf	三角形表面图
ezgraph3	函数控制绘制三维图	trimesh	三角形网格图
pie3	三维饼状图	waterfall	瀑布图
scatter3	三维散射图	cylinder	柱面图
stem3	三维离散数据图	sphere	球面图
quiver3	向量场	contour3	三维等高线

对比表 5-2 和表 5-3 可以发现其中有许多的绘图命令函数名很接近，只是在后面加了一个数字 3，它代表是绘制相应的三维图形，其实现的功能和调用方法与对应的二维绘制函数基本相同，在这里就不再做详细介绍，请读者自己查阅 MATLAB 7.0 的在线帮助。

【cylinder 命令】

该命令用于绘制圆柱图形。调用格式如下：

```
[X,Y,Z] = cylinder    //返回一半径为1、高度为1的圆柱体的x，y，z轴的坐标值，圆柱体的圆周有20个距离相同的点
[X,Y,Z] = cylinder(r)    //返回一半径为r、高度为1的圆柱体的x，y，z轴的坐标值，圆柱体的圆周有20个距离相同的点
[X,Y,Z] = cylinder(r,n)    //返回一半径为r、高度为1的圆柱体的x，y，z轴的坐标值，圆柱体的圆周有指定的n个距离相同的点
cylinder(axes_handle,...)
cylinder(...)    //没有任何的输出参量，直接画出圆柱体
```

【示例 1】

```
>>cylinder
```

用 cylinder 命令绘制圆柱图形如图 5-30 所示。

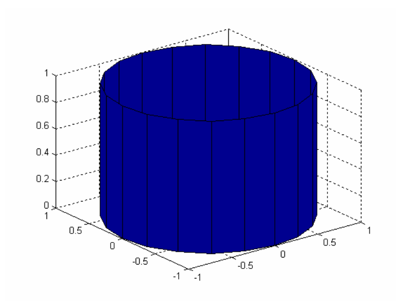


图 5-30　cylinder 命令绘制圆柱图形

【示例 2】

```
>>t = 0:pi/10:2*pi;
>>[X,Y,Z] = cylinder(2+cos(t));
>>surf(X,Y,Z)
```

用 cylinder（r）命令绘制圆柱图形如图 5-31 所示。

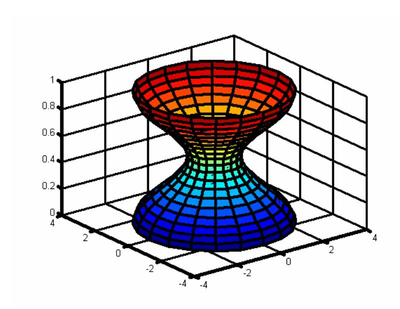


图 5-31　cylinder(r)命令绘制圆柱图

【sphere 命令】

该命令用于生成球体。调用格式如下：

```
sphere    //生成三维直角坐标系中的单位球体。该单位球体由20*20个面组成
sphere(n)    //在当前坐标系中画出有n*n个面的球体
[X,Y,Z] = sphere(...)    //返回三个阶数为(n+1)*(n+1)的，直角坐标系中的坐标矩阵。该命令没有画图，只是返回矩阵。用户可以用命令surf（x，y，z）或mesh（x，y，z）画出球体
```

【示例】

```
>>[m,n,p]=sphere(50);
>>t=abs(p);
>>surf(m,n,p,t)
```

用 sphere 命令绘制球体如图 5-32 所示。

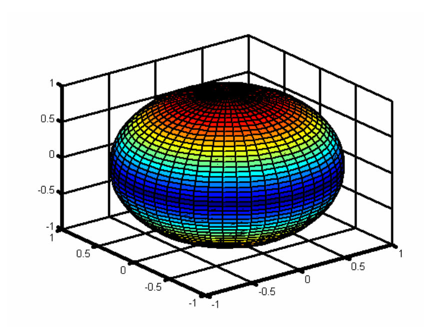


图 5-32　sphere 命令绘制球体

5.3.3　特殊坐标轴的图形函数

前面已介绍了基本的二维绘图函数的使用，但它们的坐标轴刻度均为线性刻度。当实际的数据出现指数变化时，指数变化就不能直观的从图形上体现出来，而且在进行数值比较过程中经常会遇到双纵坐标显示的要求。为了解决这些问题，MATLAB 提供了相应的绘图函数。

1．semilogx

用该函数绘制图形时 *x* 轴采用对数坐标。若没有指定使用的颜色，当所画线条较多时，semilogx 将自动使用由当前轴的 ColorOrder 和 LineStyleOrder 属性指定的颜色顺序和线型顺序来画线。调用方法如下：

semilogx(Y)　//对 *x* 轴的刻度求常用对数（以 10 为底），而 *y* 轴为线性刻度。若 *y* 为实数向量或矩阵，则结合 *y* 列向量的下标与 *y* 的列向量画出线条；若 *y* 为复数向量或矩阵，则 semilogx(Y)等价于 semilogx(real(Y),imag(Y))。在 semilogx 的其他使用形式中，Y 的虚数部分将被忽略

semilogx(X1,Y1,X2,Y2…)　//结合 Xn 和 Yn 画出线条，若其中只有 Xn 或 Yn 为矩阵，另外一个为向量，行向量的维数等于矩阵的列数，列向量的维数等于矩阵的行数，则按向量的方向分解矩阵，再与向量结合，分别画出线条

semilogx(X1,Y1,LineSpec1X2,Y2,LineSpec2,…)　//按顺序取三参数 Xn,Yn,LineSpecn 画线，参数 LineSpecn 指定使用的线型，标记符号和颜色。用户可以混合使用二参数和三参数形式，如 semilogx(X1,Y1,X2,Y2,LineSpec2,X3,Y3)

semilogx(…,'PropertyName',PropertyValue,…)　//对所有由 semilogx 命令生成的图形对象句柄的属性进行设置

h = semilogx(…)　//返回 line 图形句柄向量，每条线对应一个句柄

【示例】

```
>>x=0.001:0.01*pi:2*pi;
>>y=log10(x);
```

```
>> semilogx(x,y,'-*');
>>figure;
>>plot(x,y)
```

用 semilogx 和 plot 绘制 y=log10（x）的图形如图 5-33 和图 5-34 所示。

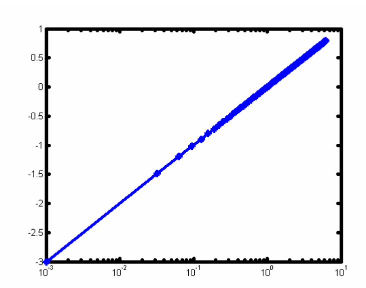


图 5-33　semilogx 绘制 y=log10(x)

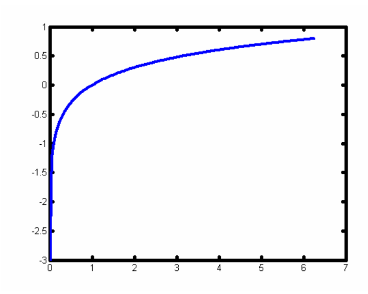


图 5-34　plot 绘制 y=log10(x)

2．semilogy

用该函数绘制图形时 *y* 轴采用对数坐标。调用格式与 semilogx 基本相同。

【示例】

```
>>x=0.001:0.01*pi:2*pi;
>>y=10.^x;
>> semilogy(x,y,'-*');
>>figure;
>>plot(x,y)
```

用 semilogy 和 plot 绘制 y=10.^x 的图形如图 5-35 和图 5-36 所示。

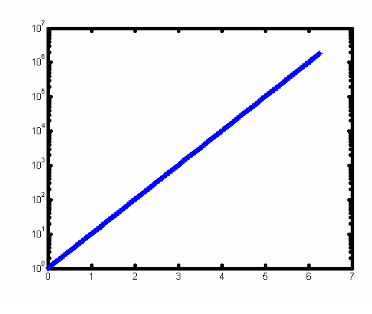


图 5-35　semilogy 绘制 y=10.^x

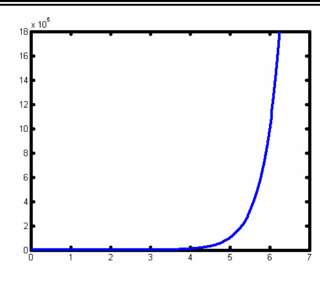


图 5-36　plot 绘制 y=10.^x

3．loglog

用该函数绘制图形时 *x* 和 *y* 轴均采用对数坐标。调用格式与 semilogx 基本相同。

【示例】

```
>>m =0.001:0.01*pi:2*pi;
>>x=10.^m;
>>y=log10(m);
>>loglog(x,y,'-*');
>>figure
>>plot(x,y,'-.r')
```

用 loglog 和 plot 绘制图形如图 5-37 和图 5-38 所示。

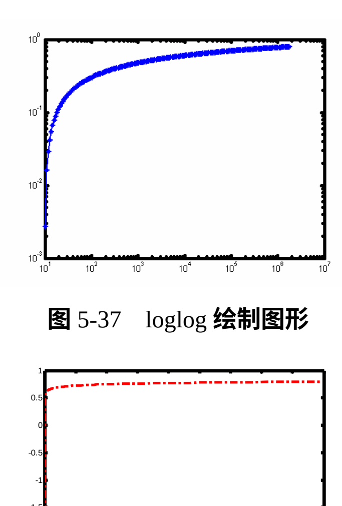


图 5-37　loglog 绘制图形

图 5-38　plot 绘制图形

4．plotyy

用该函数在双 *y* 轴坐标系中作图，调用方法如下：

```
plotyy(X1,Y1,X2,Y2)
plotyy(X1,Y1,X2,Y2,'function')   //以 function 方式绘制图形，function 可以为 plot,semilogx,semilogy 或
```

loglog 等

plotyy(X1,Y1,X2,Y2,'function1','function2') //以 function1 方式绘制(X1,Y1)，以 function2 方式绘制(X2,Y2)

【示例】

```
>>x=0:0.01*pi:2*pi;
>>y=sin(x);
>>z=exp(x);
>>plotyy(x,y,x,z,'plot','semilogy')
```

用 plotyy 绘制图形如图 5-39 所示。

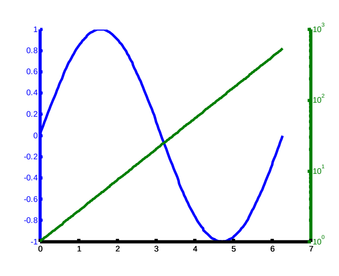


图 5-39 plotyy 绘制图形

5．极坐标、柱坐标和球坐标下绘制图形

（1）在极坐标系中绘图

【polar 命令】

该命令用于画极坐标图。它接受极坐标形式的函数 rho=f(θ)，在笛卡尔坐标系平面上画出该函数，且在平面上画出极坐标形式的格栅。调用格式如下：

polar(theta,rho) //用极角 theta 和极径 rho 画出极坐标图形。极角 theta 为从 x 轴到半径的单位为弧度的向量，极径 rho 为各数据点到极点的半径向量

polar(theta,rho,LineSpec) //参量 LineSpec 指定极坐标图中线条的线型、标记符号和颜色等

【示例】

```
>>rho0=1;
>>theta=0:pi/20:4*pi;
>>rho=rho0+theta*rho0;
>>polar(theta,rho,':')
```

用 polar 命令绘制渐开线，如图 5-40 所示。

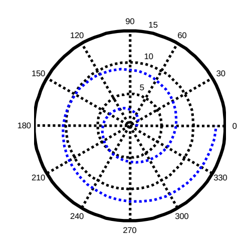


图 5-40 polar 命令绘制图形

（2）在柱坐标系中绘图

【pol2cart 命令】

该命令用于将极坐标值或柱坐标值转换成直角坐标系下的坐标值，其转换规则如图 5-41 所示。然后使用 plot3、mesh 等命令绘图，即在直角坐标系下绘制使用柱坐标值描述的图形。调用格式如下：

```
[X,Y] = pol2cart(THETA,RHO)
[X,Y,Z] = pol2cart(THETA,RHO,Z)
```

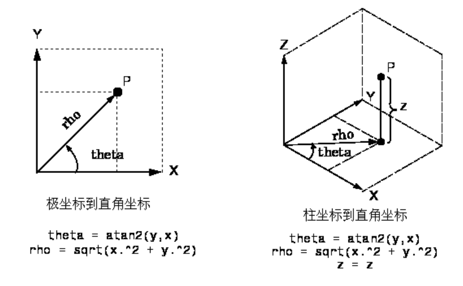


图 5-41　极坐标和柱坐标与直角坐标转换的规则

【示例】

```
>>theta=0:pi/20:2*pi;
>>rho=sin(theta);
>>[t,r]=meshgrid(theta,rho);
>>z=r.*t;
>>[X,Y,Z]=pol2cart(t,r,z);
>>mesh(X,Y,Z)
```

用 pol2cart 命令绘制出的图形如图 5-42 所示。

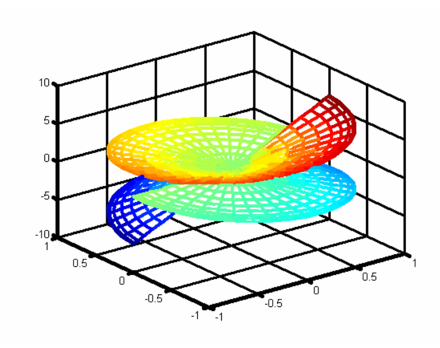


图 5-42　pol2cart 命令绘制图形

（3）在球坐标系中绘图

【sph2cart 命令】

该命令用于将球坐标值转换成直角坐标系下的坐标值，其转换规则如图 5-43 所示。然后使用 plot3、mesh 等命令绘图，即在直角坐标系下绘制使用球坐标值描述的图形。调用格式如下：

```
[x,y,z] = sph2cart(THETA,PHI,R)
```

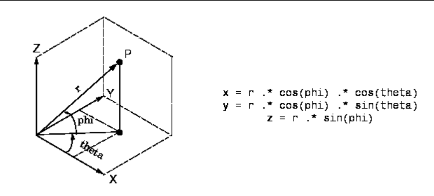


图 5-43　柱坐标与直角坐标转换的规则

（4）不同坐标系之间的转换

【cart2pol 命令】

该命令用于将直角坐标系下的坐标值转换成极坐标系和柱坐标系下的坐标值，转换规则如图 5-41 所示。其调用格式如下：

```
[THETA,RHO,Z] = cart2pol(X,Y,Z)
[THETA,RHO] = cart2pol(X,Y)
```

【cart2sph 命令】

该命令用于将直角坐标系下的坐标值转换成球坐标系下的坐标值，转换规则如图 5-43 所示。其调用格式如下：

```
[THETA,PHI,R] = cart2sph(X,Y,Z)
```

5.3.4　四维表现图

对于三维图形，可以利用 z=z（x，y）的确定或不确定的函数关系来绘制图形，但此时自变量只有两个，也即二维的。当自变量有 3 个时，定义域是整个三维空间，而我们所处的空间和思维局限性，在计算机的屏幕上只能表现出三个空间变量，因此再没有什么空间变量来表示函数的值了。为此，MATLAB 通过颜色来表示这存在于第四维空间的值，它由函数 slice 实现。其调用格式如下：

slice(V,sx,sy,sz)　//显示三元函数 V=V(X,Y,Z)确定的超立体形在 *x* 轴、*y* 轴与 *z* 轴方向上的若干点（对应若干平面）的切片图，各点的坐标由数量向量 sx、sy 与 sz 指定。其中 V 为三维数组（阶数为 m*n*p），默认的有：X = 1:m、Y = 1:n、Z = 1:p

slice(X,Y,Z,V,sx,sy,sz)　//显示三元函数 V=V(X,Y,Z)确定的超立体形在 *x* 轴、*y* 轴与 *z* 轴方向上的若干点（对应若干平面）。即若函数 V=V(X,Y,Z)中有一变量如 X 取一定值 X0，则函数 V=V(X0,Y,Z)变成一立体曲面（只不过是将该曲面通过颜色表示高度 V,从而显示于一平面而已）的切片图，各点的坐标由参量向量 sx、sy 与 sz 指定。参量 X、参量 Y 与参量 Z 为三维数组，用于指定立方体 V 的坐标。参量 X、Y 与 Z 必须有单调的、正交的间隔（如同用命令 meshgrid 生成的一样）。在每一点上的颜色由对超立体 V 的三维内插值确定

slice(V,XI,YI,ZI)　//显示由参量矩阵 XI、YI 与 ZI 确定的超立体图形的切面图。参量 XI、YI 与 ZI 定义了一个曲面，同时会在曲面的点上计算超立体 V 的值。参量 XI、YI 与 ZI 必须为同型矩阵

slice(X,Y,Z,V,XI,YI,ZI)　//沿着由矩阵 XI、YI 与 ZI 定义的曲面画穿过超立体图形 V 的切片

slice(...,'method')　//指定内插值的方法。'method'为如下方法之一：'linear'、'cubic'、'nearest'：'linear'——指定使用三次线性内插值法（该状态为默认的）；'cubic'——指定使用三次立方内插值法；'nearest'——指定使用最近点内插值法

slice(axes_handle,...)

h = slice(...)　//返回一曲面图形对象的句柄向量 h

【示例】

在-2≤x≤2，-2≤y≤2，-2≤z≤2 **区域内可视化函数** $v = xe^{(-x^2-y^2-z^2)}$。

```
>>[x,y,z] = meshgrid(-2:.2:2,-2:.25:2,-2:.16:2);
>>v = x.*exp(-x.^2-y.^2-z.^2);
>>xslice = [-1.2,.8,2]; yslice = 2; zslice = [-2,0];
>>slice(x,y,z,v,xslice,yslice,zslice)
>>colormap hsv
```

用 silice **命令绘制出的切片图如图** 5-44 **所示。**

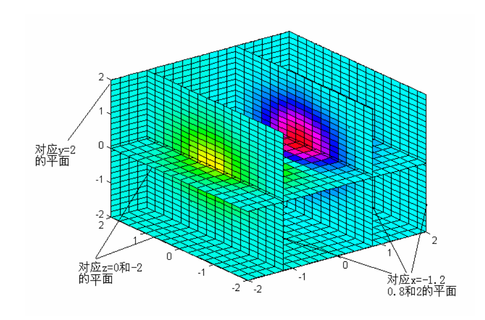


图 5-44　silice **绘制切片图**

若要显示颜色和所代表数值（在上例中即指 V **的函数值）的关系，可以通过** colorbar **函数命令来显示。**

5.4　图形处理

MATLAB 7.0 **除了提供强大的绘图功能外，它还提供了强大的图形处理的功能。下面对这些相关的技术进行具体的介绍。**

5.4.1　图形标注

从前面的示例图形可以看出，在绘制图形时，系统会自动为图形进行简单的标注。 MATLAB **语言还提供了丰富的图形标注函数供用户自由标注所绘的图形。**

1．坐标轴标注和图形标题

对坐标轴进行标注和给图形填加标题的函数主要有 xlabel、ylabel、zlabel **和** title **等，它们的调用格式基本相同，具体代码如下：**

```
xlabel('string')
xlabel(...,'PropertyName',PropertyValue,...)
xlabel(fname)
ylabel('string')
ylabel(fname)
zlabel('string')
```

```
zlabel(fname)
title('string')
title(...,'PropertyName',PropertyValue,...)
title(fname)
```

其中 string 是标注所用的说明语句，fname 是一个函数名，系统要求该函数必须返回一个字符串作为标注语句。'PropertyName',PropertyValue 用于定义相应标注文本的属性和属性值，包括字体大小、字体名和字体粗细等。

【示例】

```
>>x=1:0.1*pi:2*pi;
>>y=sin(x);
>>plot(x,y)
>>xlabel('x(0-2\pi)','fontweight','bold');
>>ylabel('y=sin(x)','fontweight','bold');
>>title('正弦函数','fontsize',12,'fontweight','bold','fontname','隶书')
```

通过上述示例代码实现对坐标轴和图形的标注，如图 5-45 所示。

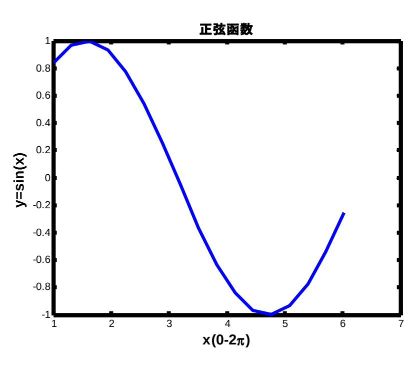


图 5-45　坐标轴标注示例

在标注过程中经常会遇到特殊符号的输入问题，如上例中 pi 的输入，MATLAB 语言提供了相应的字符转换。常见的转换如表 5-4 所示。

表 5-4　MATLAB7.0 中常见的字符转换

控制字符串	转换字符串	控制字符串	转换字符串
\alpha	α	\lambda	λ
\beta	β	\mu	μ
\gamma	γ	\xi	ξ
\delta	δ	\pi	π
\epsilon	ε	\omega	ω
\zeta	ζ	\tau	τ
\eta	η	\sigma	Σ
\theta	θ	\kappa	κ
\leftarrow	←	\uparrow	↑

用户还可以对文本标注文字进行显示控制，具体方式如下：

- \bf：黑体；
- \it：斜体；
- \sl：透视。
- \rm：标准形式；
- \fontname{fontname}：定义标注文字的字体；
- \fontsize{fontsize}：定义标注文字的字体大小。

2．用文本标注图形

在 MATLAB 语言中可以使用 text 或 gtext 命令对图形进行文本注释。使用 text 进行标注时需要定义用于注释的文本字符串和放置注释的位置，而使用 gtext 命令进行标注却可以使用鼠标来选择标注文字放置的位置。调用格式如下：

```
text(x,y,'string')
text(x,y,z,'string')
text(...'PropertyName',PropertyValue...)
gtext('string')
gtext({'string1','string2','string3',...})
gtext({'string1';'string2';'string3';...})
```

在定义标注放置的位置可以通过函数的计算值来确定，而且标注中还可以实时地调用返回值为字符串的函数，如 num2str 等，利用这些函数可以完成较为复杂的文本标注。

【示例】

```
>>x=1:0.1*pi:2*pi;
>>y=sin(x);
>>plot(x,y)
>>xlabel('x(0-2\pi)','fontweight','bold');
>>ylabel('y=sin(x)','fontweight','bold');
>>title('正弦函数','fontsize',12,'fontweight','bold','fontname','隶书')
>>text(3*pi/4,sin(3*pi/4),'\leftarrowsin(t) = .707','FontSize',16)
>>text(pi,sin(pi),'\leftarrowsin(t) = 0','FontSize',16)
>>text(5*pi/4,sin(5*pi/4),'sin(t) = -.707\rightarrow','HorizontalAlignment','right','FontSize',16)
```

用 text 命令标注的图形如图 5-46 所示。

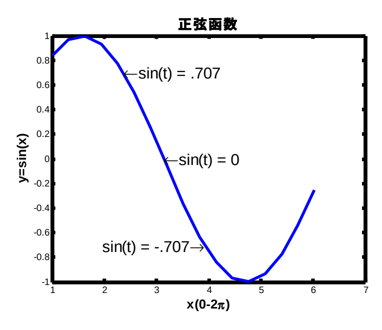


图 5-46　text 命令标注图形

【示例】

```
>>X=1:0.1*pi:2*pi;
>>Y=sin(X);
>>plot(X,Y)
>>gtext('y=sin(x) ', 'fontsize',12)
```

执行 gtext 命令后出现“ + ”形，如图 5-47 所示。执行后的结果如图 5-48 所示。

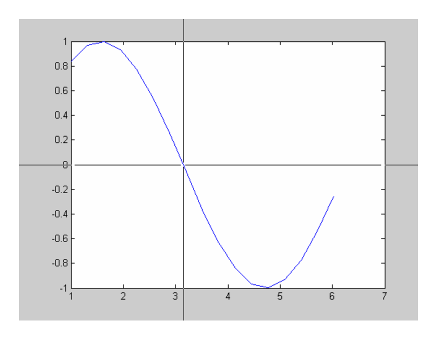


图 5-47 执行 gtext 命令后出现“ + ”型

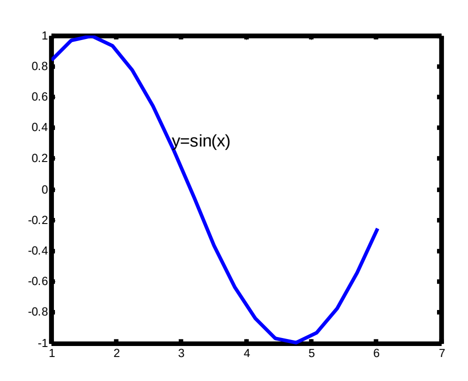


图 5-48 gtext 命令执行结果

除了上面示例中的一些标注功能外，用户还可以控制标注的对齐方式和添加多行标注文字。下面通过一个示例来说明这些功能的效果，具体的控制方法请查阅帮助。

【示例】

```
>>Z = peaks;
>>h = plot(Z(:,33));
>>x = get(h,'XData'); % Get the plotted data
>>y = get(h,'YData');
>>imin = find(min(y) == y);% Find the index of the min and max
>>imax = find(max(y) == y);
>>text(x(imin),y(imin),['Minimum=',num2str(y(imin))],'VerticalAlignment','middle','HorizontalAlignment','left','FontSize',14)
>>text(x(imax),y(imax),['Maximum=',num2str(y(imax))],'VerticalAlignment','bottom','HorizontalAlignment','right','FontSize',14)
>>str1(1) = {'Center each line in the Uicontrol'};
```

```
>>str1(2) = {'Also check out the textwrap function'};
>>str2(1) = {'Each cell is a quoted string'};
>>str2(2) = {'You can specify how the string is aligned'};
>>str2(3) = {'You can use LaTeX symbols like \pi \chi \Xi'};
>>str2(4) = {'\bfOr use bold \rm\itor italic font\rm'};
>>str2(5) = {'\fontname{courier}Or even change fonts'};
>>uicontrol('Style','text','Position',[80 80 200 30],'String',str1);
>>text(45,0,str2,'HorizontalAlignment','right')
```

通过上述示例代码来控制标注的对齐方式和添加多行标注文字，其效果如图 5-49 所示。

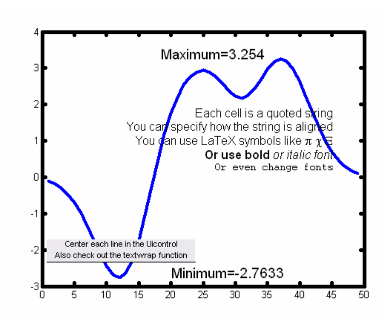


图 5-49　复杂的文本标注示例

3．图例标注

在对数值结果进行绘图时，经常会出现在一张图中绘制多条曲线的情况，这时用户可以使用 legend 命令为曲线填加图例以便区分它们。该函数能够为图形中所有的曲线进行自动标注，并以输入变量作为标注文本。其调用格式如下：

```
legend('string1','string2',...)
legend(...,'Location',location)
```

其中'string1','string2'等分别标注对应绘图过程中按绘制先后顺序所生成的曲线，'Location',location 用于定义标注放置的位置。Location 可以是一个 1×4 向量（[left bottom width height]）或任意一个字符串之一。对图例位置标注定义如表 5-5 所示。

表 5-5　图例位置标注定义

字符串	位置	字符串	位置
North	绘图区内的上中部	South	绘图区内的底部
East	绘图区内的右部	West	绘图区内的左中部
NorthEast	绘图区内的右上部	NorthWest	绘图区内的左上部
SouthEast	绘图区内的右下部	SouthWest	绘图区内的左下部
NorthOutside	绘图区外的上中部	SouthOutside	绘图区外的下部
EastOutside	绘图区外的右部	WestOutside	绘图区外的左部
NorthEastOutside	绘图区外的右上部	NorthWestOutside	绘图区外的左上部
SouthEastOutside	绘图区外的右下部	SouthWestOutside	绘图区外的左下部
Best	标注与图形的重叠最小处	BestOutside	绘图区外占用最小面积

标注的位置还可以通过定位代号来定义，代号的说明如下：

- 0：自动定位，使得标注图标与图形重叠最少；
- 1：默认值，置于图形的右上角；
- 2：置于图形的左上角；
- 3：置于图形的左下角；
- 4：置于图形的右下角；
- -1：置于图形的右外侧。

用户还可以通过鼠标来调整图例标注的位置。

【示例】

```
>>x = -pi:pi/20:pi;
>>plot(x,cos(x),'-ro',x,sin(x),'-.b')
>>h = legend('cos','sin',2);
```

通过上述示例代码可以用鼠标来调整图例标注的位置，图例标注效果如图 5-50 所示。

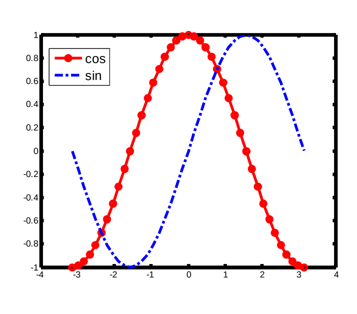


图 5-50　图例标注

5.4.2　坐标轴的控制

在 MATLAB 7.0 中可以通过设置各种参数来实现对坐标轴的控制，其中的高级控制涉及到图形句柄，这里只介绍初级的控制命令。

1．axis 函数控制坐标轴特征

该命令用于控制坐标轴的刻度范围及显示形式，调用格式如下：

```
axis([xmin xmax ymin ymax])
axis([xmin xmax ymin ymax zmin zmax cmin cmax])
axis 控制字符串'
```

其中[xmin xmax ymin ymax zmin zmax]用于定义坐标轴的范围，控制字符串可以是表 5-6 中的任一表达式。

表 5-6　axis 命令的控制字符串

字符串	说明
auto	自动模式，使得坐标轴范围能容纳下所有的图形
manual	以当前的坐标范围限定图形的绘制，此后使用 hold on 命令再次绘图时保持坐标轴范围不变
tight	将坐标范围限制在指定的数据范围内

续表

字符串	说明
fill	设置坐标范围和 PlotBoxAspectRatio 属性以使坐标满足要求
ij	将坐标设置成矩阵形式，原点在左上角
xy	将坐标设置成直角坐标系
equal	将各坐标轴的刻度设置成相同
image	与 equal 类似
square	设置绘图区为正方形
vis3d	使图形在旋转或拉伸过程中保持坐标轴的比例不变
normal	解除对坐标轴的任何限制
off	取消对坐标轴的一切设置
on	恢复对坐标轴的一切设置

【示例】

```
x = 0:.025:pi/2;
plot(x,tan(x),'-ro')
```

使用 axis 命令设定坐标轴之前的图如图 5-51 所示。

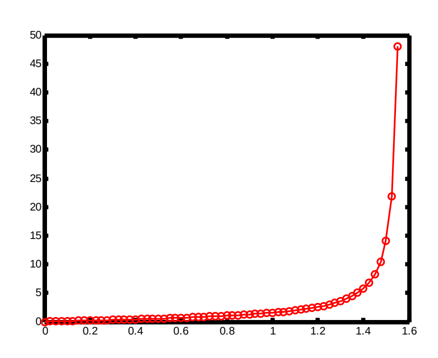


图 5-51　使用 axis 设定坐标轴之前的图

```
axis([0    pi/2    0    5])
```

使用 axis 命令设定坐标轴之后的图如图 5-51 所示。

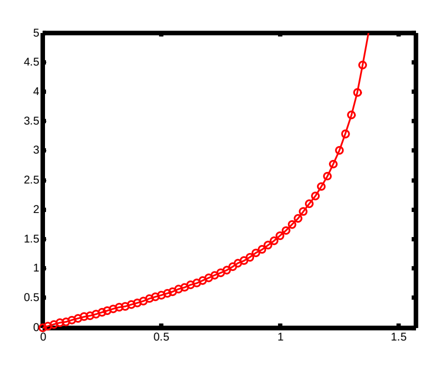


图 5-52　使用 axis 设定坐标轴之后的图

2．zoom 函数控制坐标轴缩放

该命令用于实现对二维图形的缩放，其调用格式如下：

```
zoom '控制字符串'
```

其中控制字符串可以是表 5-7 中的任一字符串。

表 5-7　　zoom 命令的控制字符串

字符串	说明
空	在 zoom on 和 zoom off 之间切换
(factor)	以 factor 作为缩放因子进行坐标轴的缩放
on	允许对坐标轴进行缩放
off	禁止对坐标轴进行缩放
out	恢复到最初的坐标轴设置
reset	设置当前的坐标轴为最初值
xon	允许对 x 轴进行缩放
yon	允许对 y 轴进行缩放

3．grid 函数控制坐标轴网格

该命令用于绘制坐标网格，其调用格式如下：

```
grid on    //给当前坐标轴填加网格线
grid off    //取消当前坐标轴的网格线
grid minor    //设置网格绘制的密度，即网格线间的间距
grid    //在 grid on 和 grid off 之间切换
```

【示例】

```
>>X=0:0.1*pi:2*pi;
>>Y=sin(X);
>>plot(X,Y);
>>grid on
```

使用 grid 函数添加网格线，如图 5-53 所示。

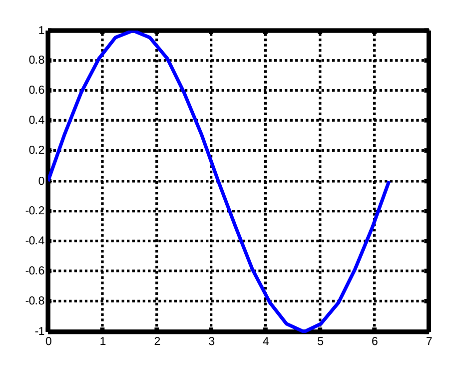


图 5-53　添加网格线

4．box 函数控制坐标轴封闭

该命令用于在图形四周都能显示坐标，其调用格式如下：

```
box on
box off
box
```

【示例】

对 grid 一节中的示例添加“>>box off”命令，结果如图 5.54 所示，注意观察与上例图形的区别。

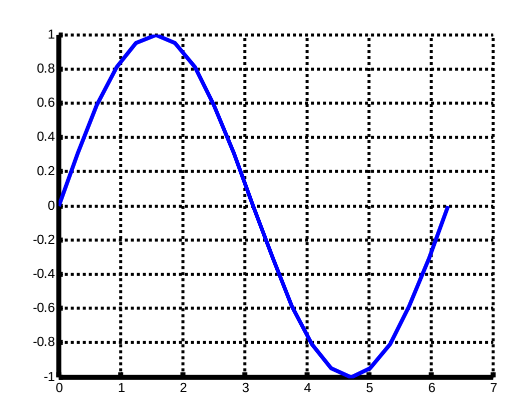


图 5-54　box 命令示例

5.4.3　图形数据取点

当用户为某一函数作好图形后，有时候会希望知道在某一自变量值下的函数值，这时使用取点命令 ginput 可方便地通过鼠标来读取二维平面图中任一点的坐标值。其调用格式如下：

```
[x,y] = ginput(n)    //用户可以通过鼠标选择 n 个点，它们的坐标值保存在[x,y]中，用户可通过 enter 键来结束取点；
[x,y] = ginput    //取点的数目不受限制，它们的坐标值保存在[x,y]中，用户通过 enter 键来结束取点；
[x,y,button] = ginput(...)    //返回值 button 记录了在选取每个点时的相关信息；
```

【示例】

```
>>x=0:0.1*pi:2*pi;
>>y=sin(x);
>>plot(x,y)
>>[m n]=ginput(1)
>>hold on
>>plot(m,n,’or’)
>>text(m(1),n(1),[’m(1)=’,num2str(m(1)),’n(1)=’,num2str(n(1))])
```

用户在通过鼠标取点前会出现“+”形，如图 5-55 所示。用 ginput 命令执行的结果如图 5-56 所示。

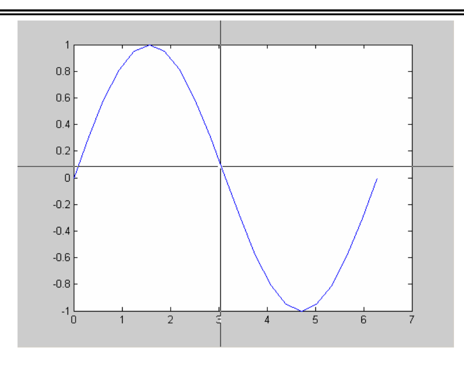


图 5-55　取点确定前出现“＋”形

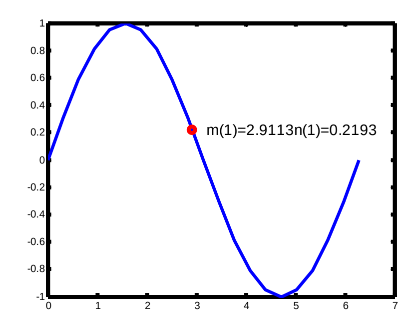


图 5-56　ginput 命令执行结果

取点结束后，在 MATLAB 7.0 命令窗口显示以下结果：

```
m =
      3.1855
n =
    -0.0322
```

5.4.4　子图和图形保持

在绘图过程中，用户会经常碰到下面两种情况。在已存的一张图中填加新的曲线；将几个图形在同一个图形窗口中表现出来，但又不在同一个坐标系中绘制。MATLAB 7.0 为这两种需要分别提供 hold 函数和 subplot 函数，前面的示例已用到 hold 函数，下面对它们的使用方法进行介绍。

【hold 命令】

该命令常用调用格式如下：

```
hold on    //启动图形保持功能，此后绘制的图形都将填加到当前的图形窗口中，并自动调整坐标轴范围
hold off   //关闭图形保持功能
hold   //在 hold on 和 hold off 之间切换
```

在前面的示例中已经使用到该命令，使用方法比较简单，用户只要实践几次就可掌握，在此不再举例说明。

【subplot 命令】

该命令用于生成并控制多个坐标轴，绘制出的效果图如图 5-57 所示。把当前图形窗口分

隔成几个矩形部分，不同的部分是按行方向以数字进行标号的。每一部分有一坐标轴，后面的图形输出在当前的部分中。常用调用格式如下：

subplot(m,n,p)　//将一图形窗口分成 m*n 个小窗口，在第 p 个小窗口中创建一坐标轴。则新的坐标轴成为当前坐标轴。若 p 为一向量，则创建一坐标轴，包含所有罗列在 p 中的小窗口。用户可以通过参数 p 分别对各子绘图区域进行操作，子绘图区域的编号按行从左至右编号，如图 5-58 所示。

subplot(m,n,p,'replace')　//如果定义的坐标轴已存在，那么就删掉已有的，并创建一个新的坐标轴

subplot(m,n,p,'align')　//对齐坐标轴

subplot(h)　//使句柄 h 对应的坐标轴称为当前的，用于后面图形的输出显示

subplot('Position',[left bottom width height])　//在由 4 个元素指定的位置上创建一坐标轴。位置元素的单位为归一化单位

【示例】

```
>>income = [3.2 4.1 5.0 5.6];
>>outgo = [2.5 4.0 3.35 4.9];
>>subplot(2,1,1); plot(income)
>>subplot(2,1,2); plot(outgo)
```

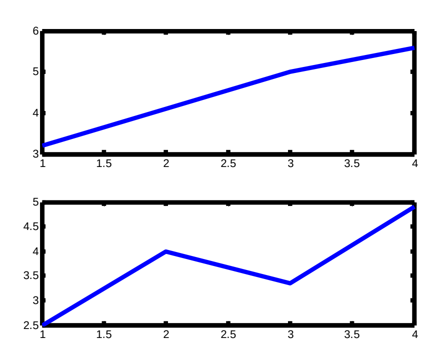


图 5-57　subplot 命令作图效果

【示例】

```
>>subplot(2,2,[1 3])
>>subplot(2,2,2)
>>subplot(2,2,4)
```

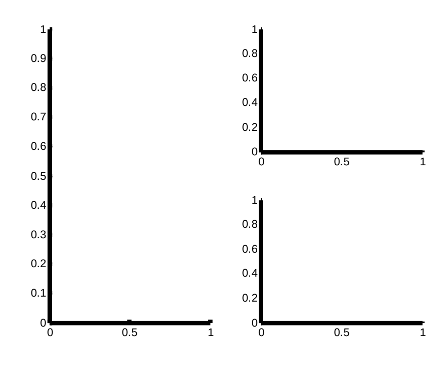


图 5-58　subplot 的 p 参数为向量

5.4.5 色彩控制

图形的一个重要因素就是图形的颜色，丰富的颜色变化能让图形更具表现力。在 MATLAB 7.0 中，色图 colormap 是完成这方面工作的主要命令。

MATLAB 是采用颜色映像来处理图形颜色的，即 RGB 色系。计算机中的各种颜色都是通过三原色按不同比例调制出来的，三原色即红（Red）、绿（Green）、蓝（Blue）。每一种颜色的值表达为一个 1×3 的向量[R G B]，其中 R、G、B 值的大小分别代表这 3 种颜色之间的相对亮度，因此它们的取值范围均必须在[0，1]区间内。每一种不同的颜色对应一个不同的向量。表 5-8 给出了典型的颜色配比方案。

表 5-8 典型的配色方案

原色			调得的颜色
红（R）	绿（G）	蓝（B）	
0	0	0	黑色
1	1	1	白色
1	0	0	红色
0	1	0	绿色
0	0	1	蓝色
1	1	0	黄色
1	0	1	洋红色
0	1	1	青色
0．5	0．5	0．5	灰色

调好颜色后，就可以用它作为绘图用色。一般的线图函数（如 plot,plot3 等）不需要色图来控制其色彩显示，而对于面图函数（如 mesh,surf 等）则需要调用色图。色图设定命令为：

```
colormap（[R，G，B]）
```

其中，输入变量[R，G，B]为一个三列矩阵，行数不限，该矩阵称为色图。表 5-9 给出了几种典型的常用色图的名称和其产生函数。

表 5-9 常用色图的名称及其产生函数

色图名称	产生函数
红黄色图	autumn
蓝色调灰度色图	bone
青红浓淡色图	cool
线性灰度色图	gray
黑红黄白色图	hot
饱和色图	hsv
粉红色图	pink
光谱色图	prism
线性色图	lines

【示例】

```
>>colormap(hsv(128))
```

此语句是用于定义当前窗口的颜色映象为饱和色图，其颜色定义了 128 种。如果用户没有定义颜色的多少，那么色图的大小（颜色种数的数目）与当前色图的大小相同。

【示例】

```
>>[x,y,z]=peaks;
>>mesh(x,y,z);
>>colormap(autumn (128))
```

使用 autumn 命令绘制的图如图 5-59 所示。

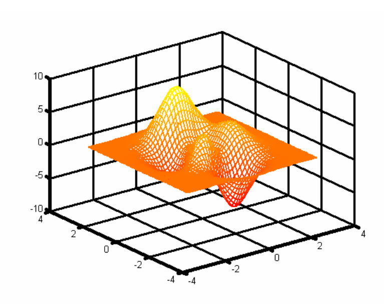


图 5-59　使用 autumn 色图绘图

除了 colormap 命令外，用户还可以通过以下一些常用函数命令对色图进行控制。

【brighten 函数命令】

该函数用于增亮或变暗色图。常见调用格式如下：

```
brighten(beta)   //增亮或变暗当前的色图。若 0<beta<1，则增亮色图；若-1<beta<0，则变暗色图。改变的色图将代替原来的色图，但本质上是相同的颜色；
brighten(h,beta)   //对指定的句柄对象 h 中的子对象进行操作；
newmap = brighten(beta)   //该命令没有改变当前图形的亮度，而是返回变化后的色图给 newmap；
newmap=brighten(cmap,beta)   //该命令没有改变指定色图 cmap 的亮度，而是返回变化后的色图给 newmap。
```

【colorbar 函数命令】

该函数用于显示能指定颜色刻度的颜色标尺。常见调用格式如下：

```
colorbar   //更新最近生成的颜色标尺。或若当前坐标轴没有任何颜色标尺，则在右边显示一垂直的颜色标尺
colorbar('vert')   //增加一垂直的颜色标尺到当前的坐标轴
colorbar('horiz')   //增加一水平的颜色标尺到当前的坐标轴
colorbar(h)   //用坐标轴 h 来生成一颜色标尺。若坐标轴的宽度大于高度，则颜色标尺是水平放置的
h = colorbar(…)   //返回一颜色标尺句柄 h，该句柄是一坐标轴对象
colorbar(…,'peer',axes_handle)   //生成一与坐标轴 axes-handle 有关的颜色标尺，代替当前的坐标轴
```

【示例】

```
>>.surf(peaks(30))
>>colorbar
```

通过 colorbar 显示颜色标尺，如图 5-60 所示。

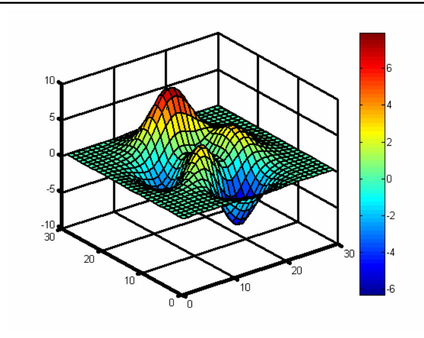


图 5-60　颜色标尺

【rgbplot 函数命令】

该函数用于画出色图。常见调用格式如下：

rgbplot(cmap)　//画出维数为 m*3 的色图矩阵 cmap 的每一列，矩阵的第一列为红色强度，第二列为绿色强度，第三列为蓝色强度

【示例】

```
>>rgbplot(copper)
```

用 rgbplot 命令绘制出的色图如图 5-61 所示。

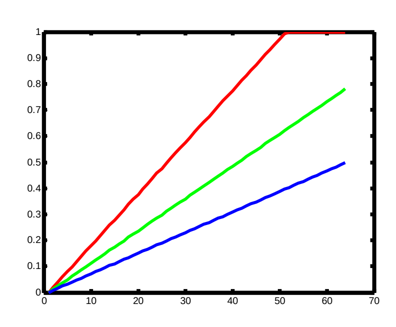


图 5-61　rgbplot 命令画出色图

【caxis 函数命令】

该函数用于颜色轴刻度，控制数值与色彩间的对应关系以及颜色的显示范围。常见调用格式如下：

caxis([cmin cmax])　//在[cmin cmax]范围内与色图的色值相对应，并依此为图形着色。若数据点的值小于 cmin 或大于 cmax，则按等于 cmin 或 cmax 来进行着色

caxis auto　//MATLAB 自动计算出色值的范围

caxis manual　//按照当前的色值范围设置色图范围

caxis(caxis)　//与 caxis manual 实现相同的功能

v = caxis　//返回当前色图的范围的最大和最小值[cmin cmax]

【示例】

```
>>.surf(peaks(30))
>>colorbar
>>caxis([-2 2])
```

用 caxis 命令绘制出的图形如图 5-62 所示。

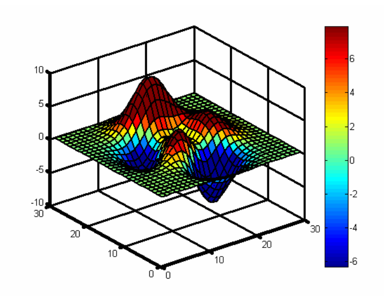


图 5-62　caxis 命令绘图效果

【shading 函数命令】

该函数用于控制曲面图形的着色方式。常见调用格式如下：

```
shading flat    //平滑方式着色
shading faceted    //以平面为着色单位，系统默认的着色方式
shading interp    //以插值形式为图形的像点着色
```

【示例】

```
>>subplot(3,1,1);    sphere(16)
>>axis square;    shading flat
>>title('Flat Shading')
>>subplot(3,1,2);    sphere(16)
>>axis square;    shading faceted
>>title('Faceted Shading')
>>subplot(3,1,3);    sphere(16)
>>axis square;    shading interp
>>title('Interpolated Shading')
```

用 shading 命令控制图形着色法方式如图 5-63 所示。

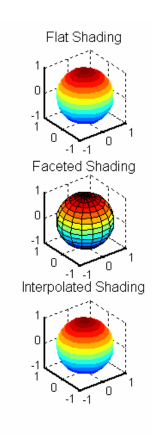


图 5-63　控制图形着色方式

【colordef 函数命令】

该函数用于设置图形的背景颜色。常见调用格式如下：

```
colordef white    //将图形背景颜色设置为白色
```

```
colordef black    //将图形背景颜色设置为黑色
colordef none    //将图形背景和图形窗口颜色设置成 MATLAB 系统的默认色彩，MATLAB7 中为黑色
colordef(fig,color_option)    //将图形句柄为 fig 的图形的背景设置为由 color_option 定义的颜色
h = colordef('new',color_option)
```

【示例】

```
>>colordef none
>>surf(peaks(30))
```

用 colordef 函数命令将图形背景色设为默认值。

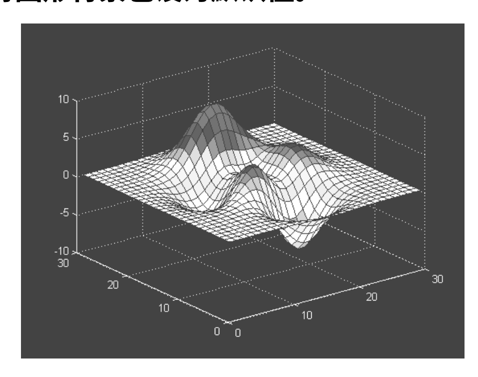


图 5-64　设置图形背景色为默认值

```
>>colordef black
>>surf(peaks(30))
```

用 colordef 函数命令将图形背景色设为黑色。

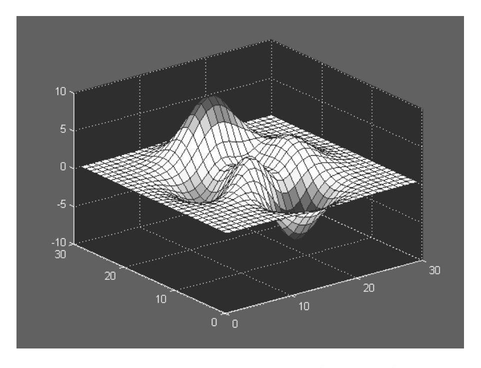


图 5-65　设置图形背景色为黑色

5.4.6　视角与光照

三维视图表现的是一个空间内的图形，因此从不同的位置和角度观察图形会有不同的效果。MATLAB 提供对图形进行视觉及光源控制的功能。所谓视觉就是图形展现给用户的角度，而光照就是图形色彩强弱变化的方向。下面向读者介绍与视角和光照相关的 MATLAB 命令。

1．视角控制命令

MATLAB 语言中用于视觉控制的命令函数主要有 view、viewmtx 和 rotate3D，下面分别

介绍它们的调用方法。

【view 函数命令】

该命令用于指定立体图形的观察点。观察者（观察点）的位置决定了坐标轴的方向。用户可以用方位角（azimuth）和仰角（elevation）一起，或者用空间中的一点来确定观察点的位置，如图 5-66 所示。其调用格式如下：

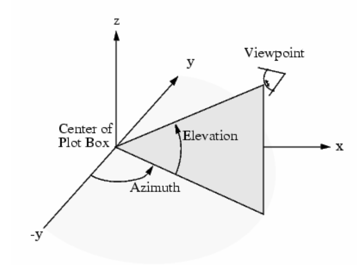


图 5-66　仰角和方位角示意图

view(az,el)、view([az,el])　//为三维空间图形设置观察点的方位角。方位角 az 与仰角 el 是按下面的方法定义的两个旋转角度：作一通过视点与 *z*-轴的平面，与 *xy* 平面有一交线，该交线与 *y*-轴的反方向的、按逆时针方向（从 *z*-轴的方向观察）计算、单位为度的夹角，就是观察点的方位角 az。若角度为负值，则按顺时针方向计算；在通过视点与 *z*-轴的平面上，用一直线连接视点与坐标原点，该直线与 *xy* 平面的夹角就是观察点的仰角 el。若仰角为负值，则观察点转移到曲面下面

view([x,y,z])　//在笛卡尔坐标系中将视角设为沿向量[x,y,z]指向原点，例如 view([0 0 1])=view(0,90)，也即在笛卡尔坐标系中将点（x,y,z）设置为视点。注意输入参量只能是方括号的向量形式，而非数学中的点的形式

view(2)　//设置默认的二维形式视点。其中 az=0，el=90，即从 *z*-轴上方观看所绘图形

view(3)　//设置默认的三维形式视点。其中 az=-37.5，el=30

view(T)　//根据转换矩阵 ***T*** 设置视点。其中 ***T*** 为 4*4 阶的矩阵，如同用命令 viewmtx 生成的透视转换矩阵一样

[az,el] = view　//返回当前的方位角 az 与仰角 el

T = view　//返回当前的 4*4 阶的转换矩阵 ***T***

【示例】

```
>>X=0:0.1*pi:2*pi;    Z=sin(X);    Y=zeros(size(X)) ;
>>subplot(2,2,1)
>>plot3(X,Y,Z,'b'); grid;
>>xlabel( 'X-axis '); ylabel( 'Y-axis '); zlabel(' Z-axis ');
>>title( ' Default Az = -37.5,E1 = 30 ');
>>view(-37.5,30);
>>subplot(2,2,2)
>>plot3(X,Y,Z);    grid;
>>xlabel( 'X-axis '); ylabel( 'Y-axis '); zlabel( 'Z-axis ')
>>title( 'Az Rotated to 52.5 ')
>>view(-37.5+90,30)
```

```
>>subplot(2,2,3)
>>plot3(X,Y,Z,'r');   grid;
>>xlabel( 'X-axis ' );   ylabel( ' Y-axis ' );   zlabel( 'Z-axis ');
>>title( ' E1 Increased to 60 ' )
>>view(-37.5,60)
>>subplot(2,2,4)
>>plot3(X,Y,Z,'g');   grid;
>>xlabel( 'X-axis ');   ylabel( ' Y-axis ');   zlabel( 'Z-axis ')
>>title( 'Az = 0,E1 = 90 ' )
>>view(0,90)
```

用 view 函数命令设置视点，效果如图 5-67 所示。

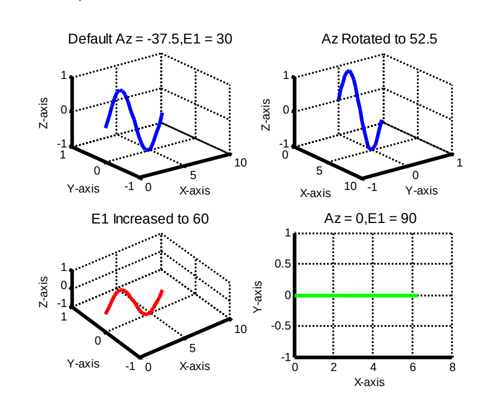


图 5-67　为图形设置视点

【viewmtx 函数命令】

该命令用于视点转换矩阵。计算一个 4*4 阶的正交的或透视的转换矩阵，该矩阵将一四维的、齐次的向量转换到一个二维的视平面上。其调用格式如下：

T = viewmtx(az,el)　//返回一个与视点的方位角 az 与仰角 el（单位都为度）对应的正交矩阵，并没有改变当前视点

T = viewmtx(az,el,phi)　//返回一个透视的转换矩阵，其中参量 phi 是单位为度的透视角度，为标准化立方体（单位为度）的对象视角角度与透视扭曲程度。Phi 的取值如表 5-10 所示。用户可以通过使用返回的矩阵，用命令 view(T)改变视点的位置

T = viewmtx(az,el,phi,xc)　返回以在标准化的图形立方体中的点 xc 为目标点的透视矩阵（就像相机正对着点 xc 一样），目标点 xc 为视角的中心点。用户可以用一个三维向量 xc=[xc,yc,zc]指定该中心点，每一分量都在区间[0，1]上。默认值为 xc=[0 0 0]

表 5-10　　phi 的取值

Phi 的值	说明
0 度	正交投影
10 度	类似以远距离投影
25 度	类似于普通投影
60 度	类似以广角投影

【示例】

```
>>x = [0   1   1   0   0   0   1   1   0   0   1   1   1   1   0   0];
>>y = [0   0   1   1   0   0   0   1   1   0   0   0   1   1   1   1];
>>z = [0   0   0   0   0   1   1   1   1   1   1   0   0   1   1   0];
>>A = viewmtx(-37.5,30);
>>[m,n] = size(x);
>>x4d = [x(:),y(:),z(:),ones(m*n,1)]';
>>x2d = A*x4d;
>>x2 = zeros(m,n);    y2 = zeros(m,n);
>>x2(:) = x2d(1,:);
>>y2(:) = x2d(2,:);
>>plot(x2,y2)
```

用 viewmtx 函数绘制出的效果图如图 5-68 所示。

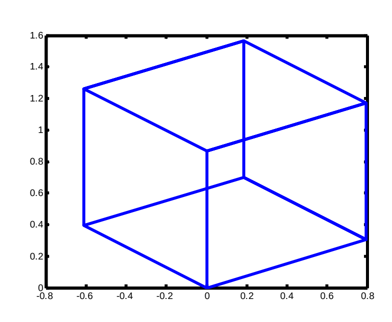


图 5-68　viewmtx 命令

【rotate3D 函数命令】

该命令为三维视角变化函数，它的使用将触发图形窗口的 rotate3D 选项，用户可以方便地用鼠标来控制视角的变化，且视角的变化值也将实时地显示在图中。

【示例】

```
>>surf(peaks(20));
>>rotate3D
```

在图形窗口的图形区域内按住鼠标左键不放来调节视角，用户可以很方便地从不同的角度来观察所绘得的图形，效果如图 5-69 所示。

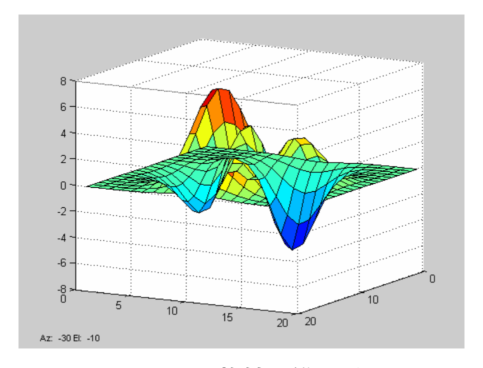


图 5-69　旋转三维图形

2．三维透视命令

在 MATLAB7.0 中使用 mesh 等命令绘制网格曲面时，系统在默认情况下会隐藏重叠在后面的网格，利用透视命令 hidden 可以看到被遮的部分。其调用格式如下：

```
hidden on
hidden off
```

【示例】

```
>>mesh(peaks)
>>hidden    off
```

用三维透视命令绘制完成的图形如图 5-70 所示。

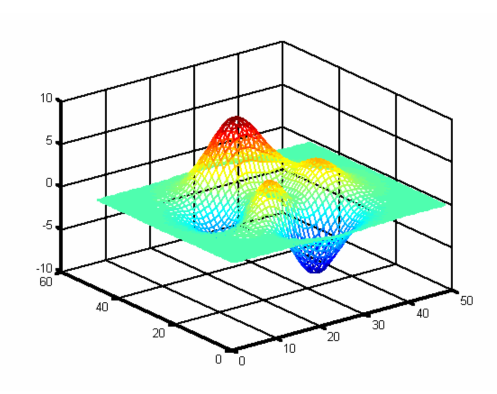


图 5-70　三维透视图形

3．光照控制命令

MATLAB 语言提供了如表 5-11 所述的光照控制命令。本小节只介绍其中几个常用的函数命令，其他命令的用法请读者查阅帮助。

表 5-11　　MATLAB 语言中的图形光照控制命令

函数名	说明	函数名	说明
light	设置曲面光源	specular	镜面反射模式
surfl	绘制存在光源的三维曲面图	diffuse	漫反射模式
lighting	设置曲面光源模式	lightangle	球坐标系中的光源
material	设置图形表面对光照反映模式		

【light 函数命令】

该命令为当前图形建立光源。其主要的调用格式如下：

```
light('PropertyName',PropertyValue,...)    // PropertyName 属性名是一些用于定义光源的颜色、位置和类型等的变量名，更为详细的介绍请查阅帮助
handle = light(...)    //返回光源对象的句柄
```

【示例】

```
>>h = surf(peaks);
>>light('Position',[1 0 0],'Style','infinite');
```

用 light 命令为图形设置光源，如图 5-71 所示。

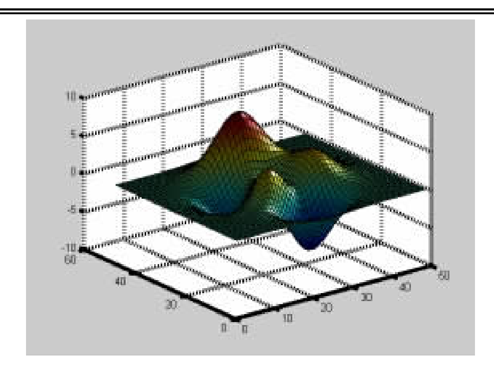


图 5-71　为图形设置光源

【lighting 函数命令】

该命令用于设置曲面光源模式，其调用格式如下：

```
lighting flat　//平面模式，以网格为光照的基本单元。这是系统默认的模式
lighting gouraud　//点模式，以像素为光照的基本单元
lighting phong　//以像素为光照的基本单元，并计算考虑了各点的反射
lighting none　//关闭光源
```

【示例】

```
>>subplot(2,2,1)
>>mesh(peaks);light('Position',[1 0 0],'Style','infinite');
>>lighting phong
>>subplot(2,2,2)
>>mesh(peaks);light('Position',[1 0 0],'Style','infinite');
>>lighting flat
>>subplot(2,2,3)
>>mesh(peaks);light('Position',[1 0 0],'Style','infinite');
>>lighting none
>>subplot(2,2,4)
>>mesh(peaks);light('Position',[1 0 0],'Style','infinite');
>>lighting gouraud
```

用 lighting 命令设置曲面光源模式，如图 5-72 所示。

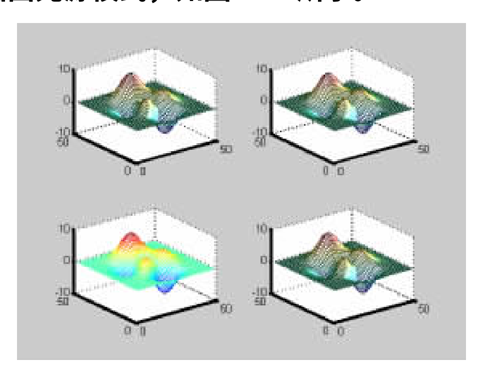


图 5-72　设置曲面光源模式

【material 函数命令】

该命令用于设置图形表面对光照反映模式，其调用格式如下：

```
material shiny    //图形表面显示较为光亮的色彩模式
material dull    //表面显示较为阴暗的色彩模式
material metal    //表面呈现金属光泽的模式
material([ka kd ks])    //[ka kd ks]用于定义图形的 ambient/diffuse/specular 三种反射模式的强度
material([ka kd ks n])    // n 用于定义镜面反射的指数
material([ka kd ks n sc])    // sc 用于定义镜面反射的颜色
material default
```

【surfl 函数命令】

该命令用于绘制三维带光照模式的阴影图。surfl(…)效果与命令 surf(…)基本上一样，但是它会受光源的影响。图形的色泽取决于曲面的漫反射、镜面反射与环境光照模式。其主要调用格式如下：

surfl(Z)、surfl(X,Y,Z)、surfl(Z,S)、surfl(X,Y,Z,S)、surfl(X,Y,Z,S,K)   //这些都是有效的使用形式。参数 S 为一个三维向量[Sx,Sy,Sz]，用于指定光源的方向。S 也可视为点坐标系下的二维向量[AZ,EL]。S 的默认值为从当前观察方向逆时针旋转 45°。使用命令组 cla；hold on；view(AZ,EL)；surfl(…)；hold off 等可画出视角方向为(AZ,EL)的带光照模式的曲面图形。第 5 个参数 K=[ka,kd,ks,spread]用于指定环境光、漫反射光、镜面反射光、扩散系数等的强弱

surfl(…,'light')   //用 LIGHT 对象生成一带颜色和光照模式的曲面。该命令可以生成与用默认光照模式效果不同的曲面

surfl(…,'cdata')   //指定的曲面的反射光的颜色为 cdata

H = surfl(…)   //返回曲面与光源的句柄

【示例】

```
>>[X,Y] = meshgrid(-3:1/8:3);
>>Z = peaks(X,Y);
>>surfl(X,Y,Z);
>>shading    interp;
>>colormap(gray);
```

用 surfl 命令绘制的图形如图 5-73 所示。

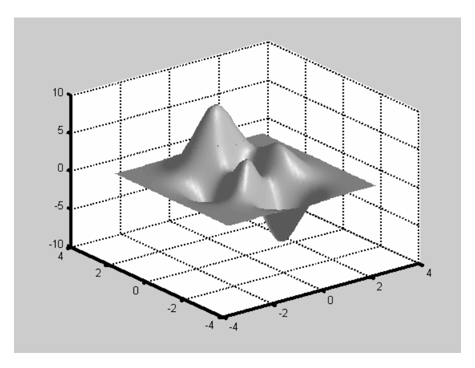


图 5-73   surfl 函数绘图

5.4.7 图形的打印和输出

用户在使用各种绘图命令作出精美的图形后，经常需要将图形打印或是以图片形式放在其他文档中保存或是在其他图片处理软件中作进一步的处理。MATLAB 为用户提供了 3 种不同的方式输出当前的图形。首先是通过图形窗口的命令菜单或是工具栏中的打印选项来输出；其次，使用 MATLAB 语言提供的内置打印引擎或系统的打印服务来实现；最后可以以其他的图形格式存储图形，然后通过其他的图形处理软件对其进行处理和打印。

利用菜单或工具栏来实现打印非常简单，这也是用户最常用的一种方式，在这里不作详细介绍。下面简单介绍实现打印的函数 print 的基本调用格式。代码设置如下：

```
print    //将图形发送到由 printopt 定义的打印设备和系统打印命令中
print  filename    //将图形输出到文件 filename，如果 filename 没有扩展名，print 命令将自动选择一个合适的扩展名
print –ddriver    //使用由 ddriver 定义的打印设备打印当前图形
print  –dformat    //将当前图形复制到系统粘贴板（仅对 Windows 适用），图片的格式可以是-dmeta (Windows Enhanced Metafile)或 dbitmap (Windows Bitmap).
print -dformat filename    //以用户自定义的图形格式 dformat 将图形输出到用户自定义的 filename 文件中
print –smodelname    //打印当前 Simulink 模型 smodelname
print ... –options    //定义打印选项，关于各种可设置的选项名请读者查阅帮助
[pcmd,dev] = printopt    //返回当前系统的打印命令到字符串变量 pcmd 和输出设备到 dev 字符串变量
```

5.5 图形窗口

用户通过前面介绍的绘图命令得到的图形都是在相同的图形窗口中绘制的，它们都具有相同的窗口菜单和工具栏。这个窗口是 MATLAB 7.0 所有图形输出的专用窗口，在利用前面的每一个绘图命令绘图时，这个窗口是随之自动生成的，所以用户到此为止可能并不知道这个窗口是如何产生的，以及它有哪些作用。下面将向读者介绍如何在利用函数命令绘制图形之前生成图形窗口，以及利用窗口中的菜单和工具栏命令对图形和图形窗口的属性进行各种设置。

5.5.1 图形窗口的创建与控制

创建图形窗口的命令是 figure，其调用方式如下：

```
figure    //创建一个图形窗口对象
figure('PropertyName',PropertyValue,...)    //按照用户自定义的属性创建一个图形窗口对象，用户可以为图形窗口对象设置的属性和参数请查阅帮助
figure(h)    //如果 h 句柄所对应的窗口对象已存在，则该命令使得该图形窗口成为当前窗口；如果不存在，则新建一个句柄值为 h 的窗口对象
h = figure(...)    //返回图形窗口对象的句柄
```

用户可以通过下面两个命令来分别查阅和设置图形窗口的属性和参数。

```
get(n)   //返回句柄值为 h 的图形窗口的参数名称及其当前值
set(n)   //返回可为句柄值为 h 的图形窗口的参数名称及其用户可为这些参数设置的值
```

【示例】

- 在 MATLAB7.0 命令窗口执行“>>figure(2)”命令，打开“Figure2”窗口，如图 5-74 所示。

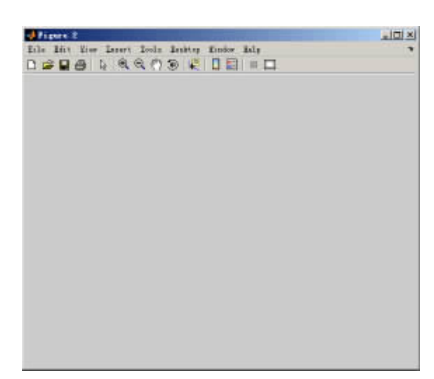


图 5-74　创建图形窗口

> 注意：图形窗口的标题为 Figure 2，如果使用“>>figure”命令则标题为 Figure 1。

- 执行“>>get(2)”，则在命令窗口中出现下面的代码。

```
Alphamap = [ (1 by 64) double array]
BackingStore = on
CloseRequestFcn = closereq
Color = [0.8 0.8 0.8]
Colormap = [ (64 by 3) double array]
...... //为了减小篇幅删去了这一部分内容
UserData = []
Visible = on
```

- 执行“>>set(2)”，则在命令窗口中出现下面的代码。

```
Alphamap
BackingStore: [ {on} | off ]
CloseRequestFcn: string -or- function handle -or- cell array
Color
Colormap
......//为了减小篇幅删去了这一部分内容
UIContextMenu
UserData
Visible: [ {on} | off ]
```

5.5.2　图形窗口的菜单操作

下面详细介绍图形窗口的各项菜单命令。

1．【File】菜单

该菜单与 Windows 系统的菜单类似，下面介绍其各个菜单项的功能。

【New】选项用于新建一个 M-文件（M-File）、图形窗口（Figure）、Simulink 模型（Model）、MATLAB 7.0 工作空间的变量（Variable）或用户界面（GUI）。其中创建 GUI 设置的对话框如图 5-75 所示。

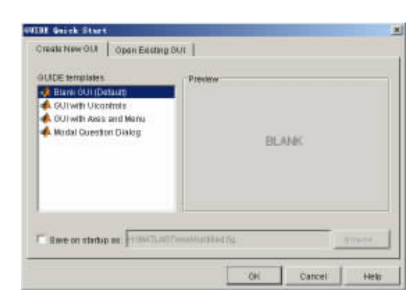


图 5-75　创建 GUI 的设置对话框

【Generate M-File】选项用于生成 M-函数文件。

【示例】

- 在 MATLAB 7.0 命令窗口中执行“>>mesh(peaks)”命令，打开如图 5-76 所示的窗口。

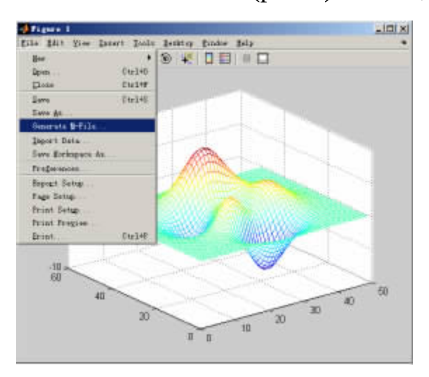


图 5-76　绘制图形

- 在生成的图形窗口中选择“Generate M-File”选项，生成如图 5-77 所示的文件，保存该文件。

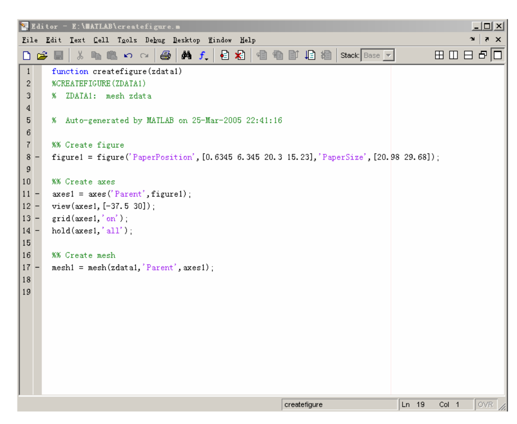


图 5-77　生成 M-函数文件

- 在 MATLAB 命令窗口执行“>>createfigure(peaks)”命令可以得到与 mesh(peaks)命令一样的图形（包括对图形的各种设置也是一样的），如图 5-76 所示。

【Import Data】选项用于导入数据。

【Save Workspace As】选项用于将图形窗口中的图形数据存储在二进制 mat 文件中，它们可以供其他的编程语言（如 C 语言等）调用。

【Preferences】选项用于定义图形窗口的各种设置，包括字体、颜色等，如图 5-78 所示。

【Export Setup】选项用于打开“图形输出”对话框，可以把图形以 emf、ai、bmp、eps、jpg、pdf 等格式保存，并设置有关图形窗口的显示等方面的参数，如图 5-79 所示。读者可以选择不同的设置以观察图形窗口有什么变化。

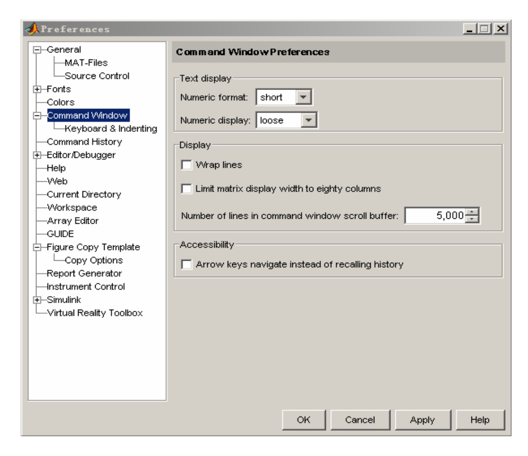


图 5-78 “图形窗口设置”对话框

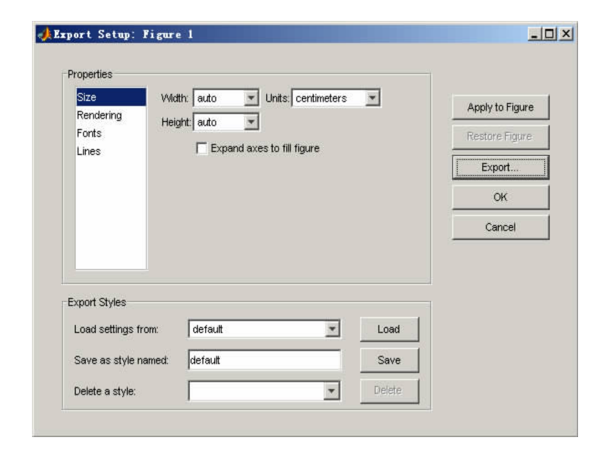


图 5-79 “图形输出”对话框

【Page Setup】选项用于打开“页面设置”对话框，如图 5-80 所示。设置图形尺寸、纸张大小、线型及文本类型以及坐标轴和图形设置。

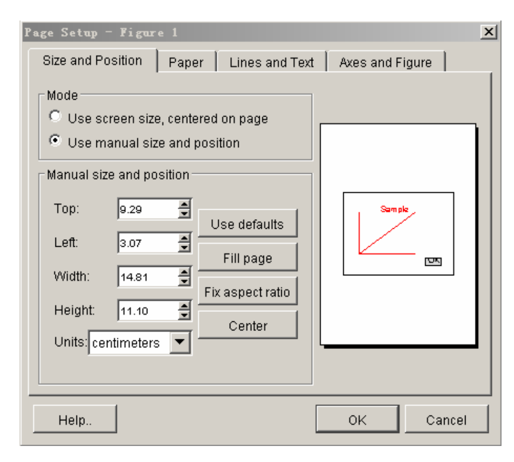


图 5-80 “页面设置”对话框

【Print Setup】选项用于打开“打印设置”对话框。在这里可以设置图片的题图等。

【Print Preview】选项用于打开打印预览对话框，如图 5-81 所示。

【Print】选项用于打开打印对话框。

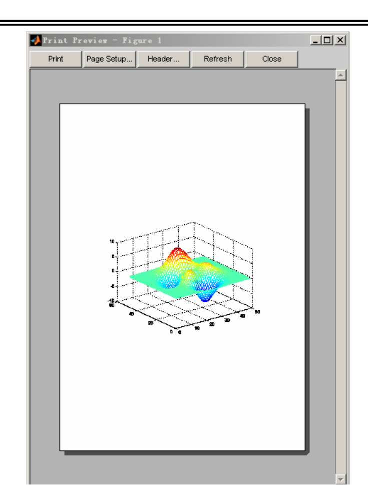


图 5-81　“打印预览”对话框

2．【Edit】菜单

该菜单中各选项的主要功能如下。

【Copy Figure】选项用于复制图形。

【Copy Option】选项用于打开“复制设置”对话框，设置图形复制的格式、图形背景颜色和图形大小等。该选项打开的对话框界面与“File→Preferences”的界面相同，只是当前显示的面板不同。

【Figure Properties】选项用于打开图形窗口的属性设置对话框，如图 5-82 所示。

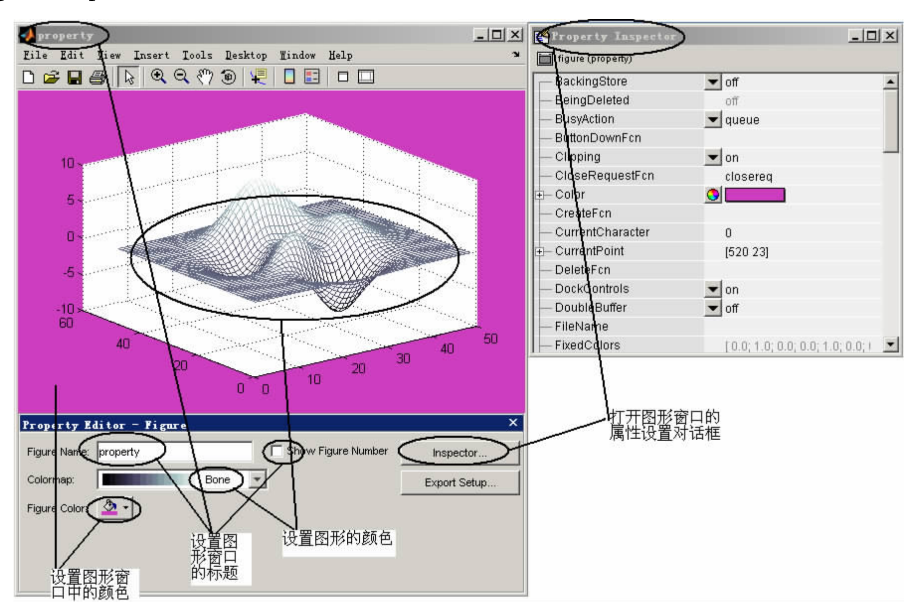


图 5-82　“图形窗口的属性设置”对话框

【Axes Properties】选项用于打开“设置坐标轴属性”对话框。

【Current Object Properties】选项用于打开设置图形窗口中当前对象（如窗口中的坐标轴、图形等）属性的对话框。

【Colormap】选项用于打开“色图编辑”对话框，如图 5-83 所示。

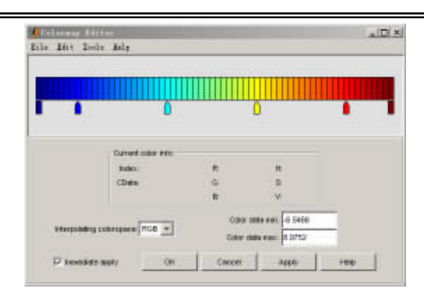


图 5-83 “色图编辑”对话框

【Clear Figure】、【Clear Command Window】、【Clear Command History】和【Clear Workspace】选项分别用于清除图形窗口中的图形、清除命令窗口、清除历史命令和清空工作空间。

3．【View】菜单

该菜单各选项主要用于打开各种工具栏和控制面板，选中该菜单的所有选项后得到如图 5-84 所示的图形窗口。

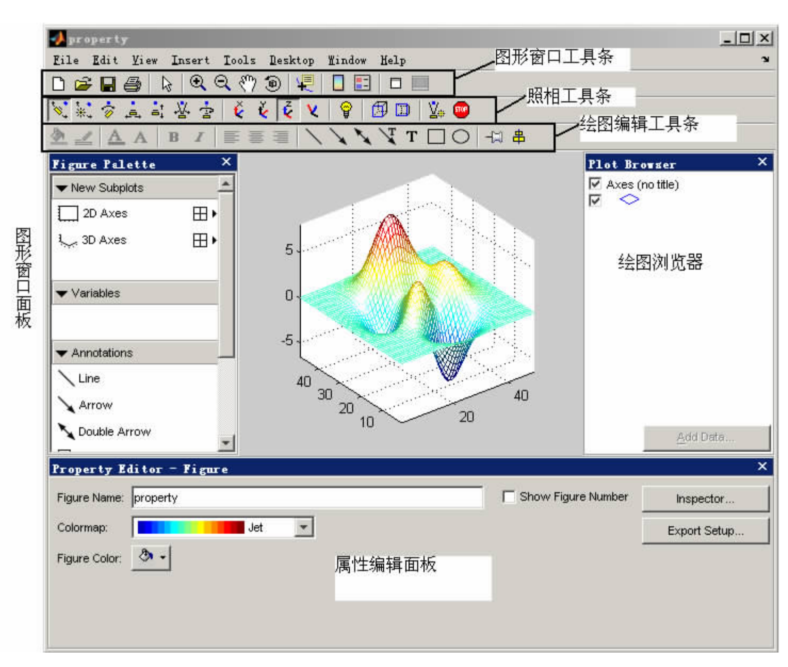


图 5-84 打开各种工具栏的图形窗口

（1）图形窗口工具条主要用于对图形进行各种处理（如旋转三维图形、取得图形中某点的坐标值、插入图例标注和插入色图条等）；

（2）照相工具条主要用于设置图形的视角和光照等，用户可以通过它们来实现从不同角度来观察所绘三维图形，并且为图形设置不同的光照情况；

（3）绘图编辑工具条主要用于向图形中添加文本标注和各种标注图形等；

（4）绘图浏览器用于浏览当前图形窗口中的所有图形对象。

读者只要对其中各种工具栏的各个工具图标加以试验，就不难发现前面介绍的许多处理

图形的命令其实可以直接通过这些直观的图标工具来实现，读者只要熟练掌握这些工具条的应用，就可以完成大部分图形处理工作，而不需要去记忆大量的命令函数。

4．【Insert】菜单

该菜单主要用于向当前图形窗口中插入各种标注图形（如插入单箭头、文字等），即实现绘图编辑工具条中的各种功能。

5．【Tools】菜单

该菜单中大部分选项实现的功能使用前面介绍的几个工具条的相关命令图标同样可以实现，这里通过一个示例来重点介绍【Basic Fitting】和【Data Statistics】两菜单项的功能。

【示例】

- 在 MATLAB 7.0 命令窗口中执行“ >>figure ”命令，然后选中【View】菜单中所有的选项，如图 5-85 所示。

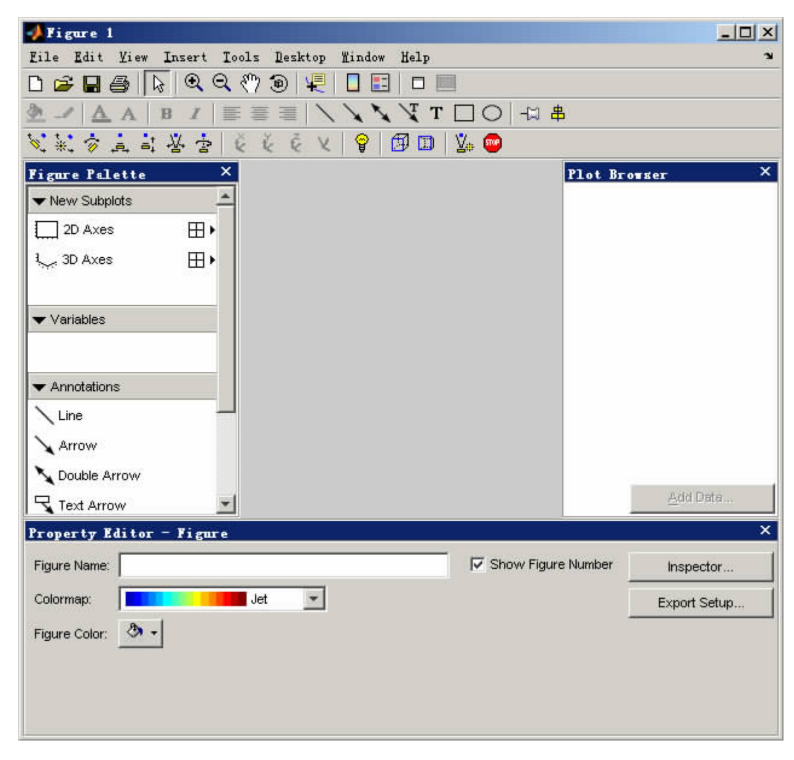


图 5-85　“ Figure ”窗口

- 在 MATLAB 命令窗口中执行下面的命令，MATLAB 的工作空间中出现两个变量 x,y。

```
>>x=[0:0.1*pi:2*pi];
>>y=sin(x)+3;
```

- 在图 5-85 中的【Figure Palette】面板中选择“ 2D Axes ”，然后选择【Plot Browser】中的“ Axes(no title) ”使该浏览器中的“ Add Data ”按钮变成可用，并单击该按钮。按如图 5-86 所示设置 x，y 轴的数据。

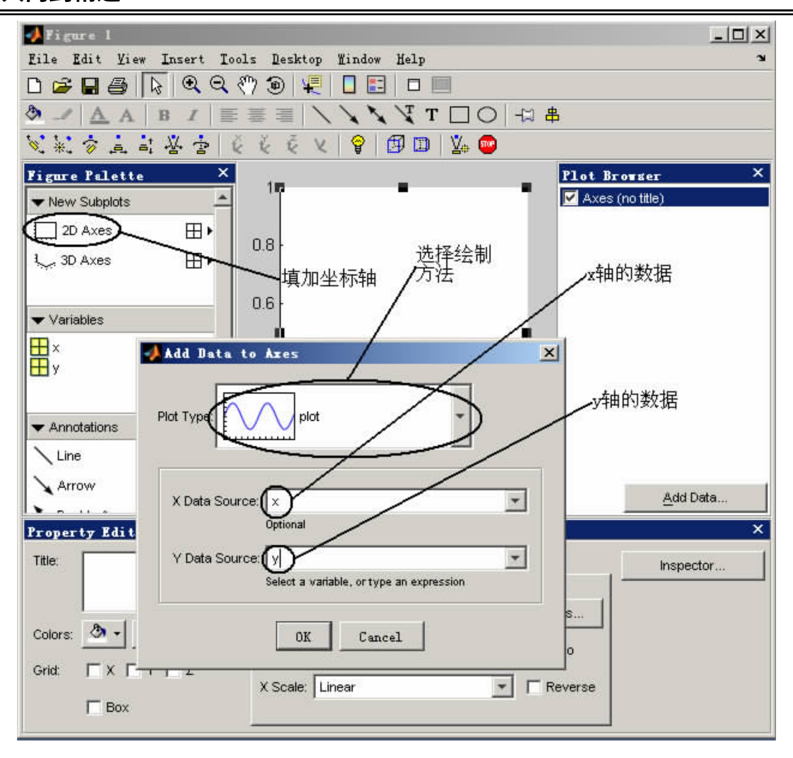


图 5-86　为图形窗口添加坐标轴和数据

- 设置好绘图方式和 x，y 轴的数据变量后，单击图 5-86 中的“OK”按钮，就会绘制出数据，如图 5-87 所示。

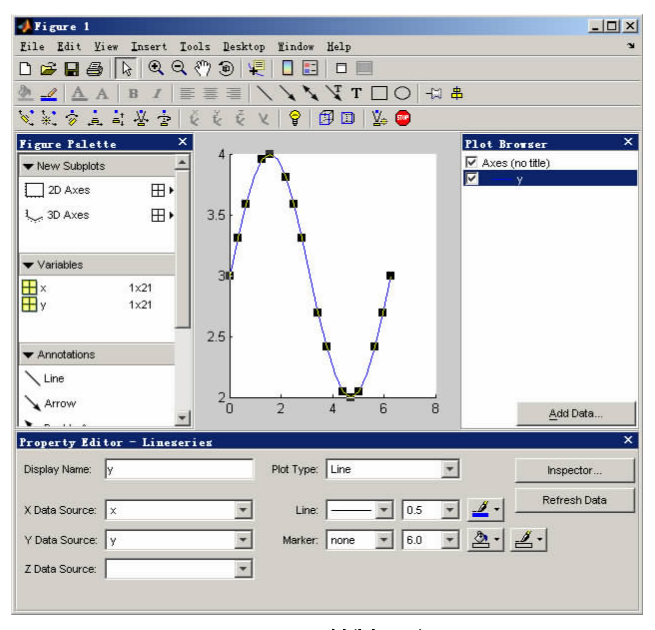


图 5-87　绘制图形

- 选中【Tools】菜单中的“Data Statistics”选项，出现如图 5-88 所示的数据统计对话框，其中显示了 x，y 变量数据的最小值、最大值、平均值等。选中 x，y 列中的“mean”选择框，观察绘图区的变化。

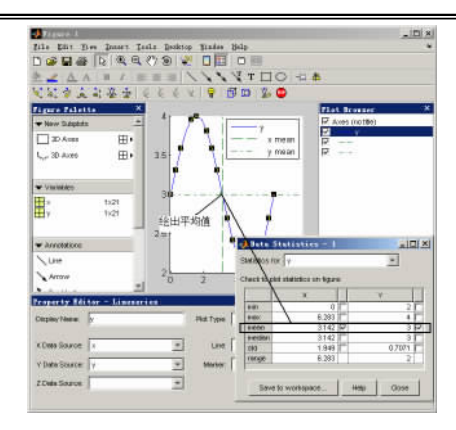


图 5-88　图形数据的统计

- **取消【View】菜单中的“Figure Palette”、“Plot Browser”和“Property Editor”选项，选中【Tools】菜单中的“Basic Fitting”选项，然后选择右箭头按钮，并按如图 5-89 所示进行设置。图形窗口就会变成如图 5-90 所示的拟合曲线和拟合误差曲线。**

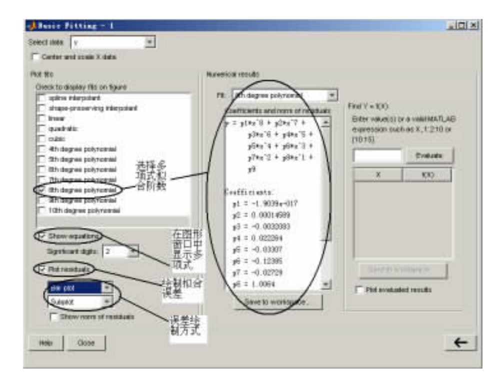


图 5-89　设置图形数据拟合的参数

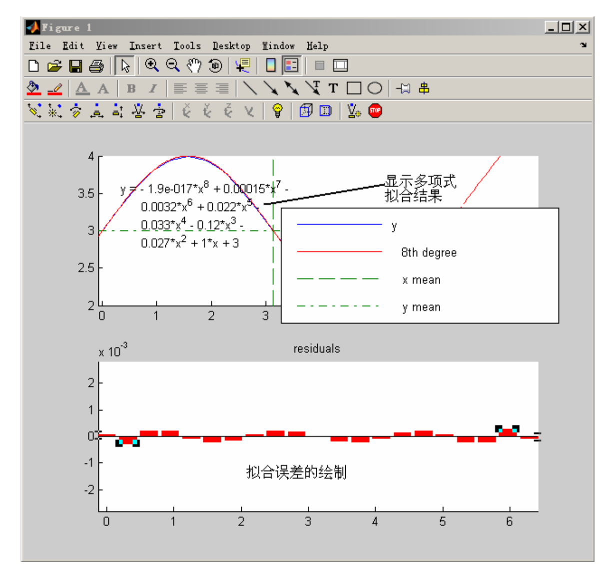


图 5-90　曲线拟合的结果

6.【Desktop】菜单

该菜单用于将窗口合并到 MATLAB 7.0 主界面的窗口中。

7.【Window】菜单和【Help】菜单

这两个菜单与 Windows 系统中各种应用程序的用户界面的相关菜单很类似，在此不作详细介绍。

5.5.3 图形窗口的工具栏

在表 5-12 中对如图 5-85 所示的 figure 窗口中各工具栏的功能进行了较为详细的介绍。

表 5-12 图形窗口工具栏

工具栏图标	说明	工具栏图标	说明
	新建一个图形窗口		打印图形
	打开图形窗口文件（后缀.fig）		移动图形
	保存图形窗口文件		旋转三维图形
	放大图形窗口中的图形		取点
	缩小图形窗口中的图形		插入颜色条
	插入图例		隐藏绘图工具
	打开绘图工具		设置环行视角
	设置光照相关属性		倾斜视角
	在水平方向设置视角		前后移动视角
	设置视角大小		水平面移动视角
	以 X 方向为标准设置环行视角		以 Y 方向为标准设置环行视角
	以 Z 方向为标准设置环行视角		选择是否打开光照
	重置图形的视角和光照		停止光照和视角的移动
	设置绘图颜色		边界颜色
	填加下画线		设置字体
	粗体		倾斜字体
	左对齐		居中
	插入直线		出入箭头
	插入双箭头		插入文本箭头
	插入文本框		插入方框
	插入椭圆		为图形上的点填加 pin
	对齐		编辑模式

【示例】

- 在命令窗口执行下面的命令：

```
>>plot([0:0.1:12],sin([0:0.1:12]))
hold on
```

```
plot([0:0.1:12],[0:0.1:12])
```

- 选中图形窗口中 View 菜单的前三项，并为图形填加一双箭头，如图 5-91 所示。

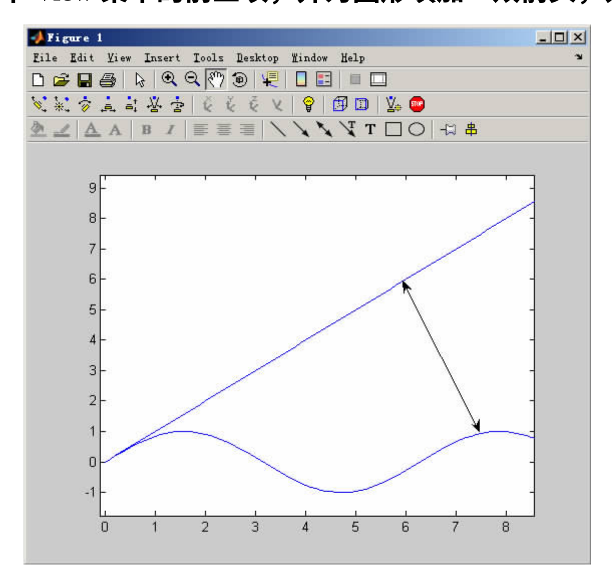


图 5-91　双箭头示例

- 选中双箭头，再单击按钮在双箭头的两个箭头处分别添加 pin，然后单击按钮移动图形，可以发现两条曲线的相对位置不变，如图 5-92 所示。

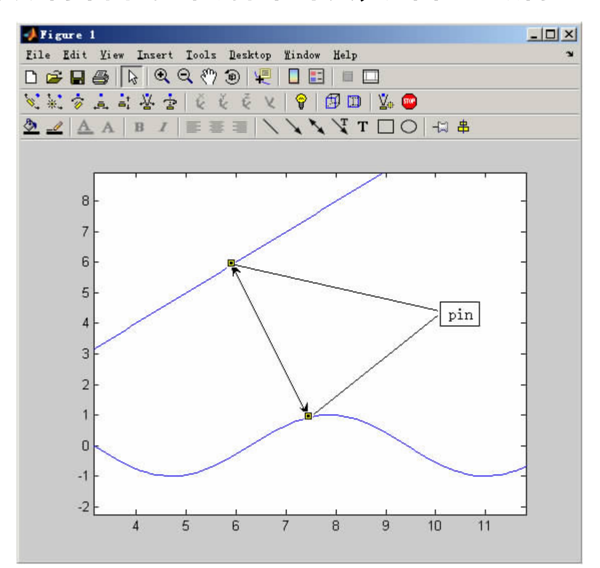


图 5-92　为图形添加 pin

第 6 章　数据分析

数据分析和处理是各种应用中非常重要的问题。针对数据分析和处理，MATLAB 7.0 提供大量的函数方便用户使用。本章将介绍 MATLAB 7.0 强大的数据分析和处理功能。

本章 6.1 小节介绍多项式函数，这些函数用于多项值求值、多项式乘法、多项式除法、多项式求导等。6.2 小节介绍插值函数。MATLAB 7.0 提供数个不同的插值算法，这些算法的性能比较将在本小节中详细介绍。6.3 小节涵盖内容比较多，包括基本的数据分析函数（例如矩阵的平均值）、协方差和相关矩阵、有限差分和梯度、信号滤波和卷积以及傅立叶变换。6.4 小节介绍功能函数，包括函数的表示方法、函数的画图、函数的最小值和最大值、函数的数值积分以及在功能函数中使用参数的方法。6.5 小节介绍了三类微分方程组问题的解法，包括常微分方程组的初值问题、延迟微分方程组数值解和常微分方程组的边界问题。

6.1　多项式函数

MATLAB 7.0 提供了一些处理多项式的基本函数，如求多项式的值、多项式的根和多项式的微分等。另外，MATLAB 7.0 也提供了一些高级函数处理多项式，如曲线拟合和多项式的部分分式表示。这些函数保存在 MATLAB 7.0 工具箱的 polyfun 子目录下，如表 6-1 所示。

表 6-1　多项式函数

函数名	功能描述
conv	多项式乘法
deconv	多项式除法
poly	求多项式的系数
polyfit	多项式曲线拟合
polyder	求多项式的一阶导数
polyint	求多项式的积分
polyvar	求多项式的值
polyvarm	求矩阵多项式的值
residue	部分分式展开
roots	求多项式的根

此外在 MATLAB 7.0 的符号工具箱（Symbolic Math Toolbox）中还有一些是用于处理多项式的函数，在表 6-1 中没有提到。

6.1.1　多项式表示法

MATLAB 7.0 采用行向量来表示多项式，将多项式的系数按降幂次序存放在行向量中。多项式 $P(x)=a_0x^n+ a_1x^{n-1}+\dots + a_{n-1}x+a_n$ 的系数行向量为 $P=[a_0\ a_1\ \dots\ a_n]$，注意顺序必须是从高次幂到低次幂。例如，多项式 x^4+3x^3+4x+5 可以用系数向量[1 3 0 4 5]来表示，注意多项

式中缺少的幂次要用“ 0 ”补齐。

例如，设计一个函数 poly2str()，实现把一个行向量表示的多项式转换为常见的字符串表示的多项式，代码设置如下：

```
%poly2str.m
%把多项式的行向量表示转换为字符串表示
function Y=poly2str(X)
%X 是表示一个多项式的向量
%Y 多项式的字符串表示
%输入检查，如果 X 不是一个向量则退出
if isvector(X)==0,
    disp('输入错误：输入 X 不是一个向量，请输入一个代表多项式的向量！');
    return;                         %函数返回
end;
Y='';                               %输出字符串
n=length(X);
for i=1:n,                          %把多项式的每一次幂转换为字符串
    if(i~=1&&X(i)>0)                %如果是正系数，必须添加‘ + ’字符
        Y=[Y '+'];
    end;
                                    %输出系数
    if(X(i)==0),                    %如果该次幂系数为 0，则不输出字符串
        continue;
    elseif(X(i)==1&&i~=n),          %如果该次幂系数为 1，可以不输出系数，只输出 x^n
        Y=Y;
    else
        Y=[Y num2str(X(i))];        %其他情况输出系数
    end;
                                    %输出 x^n
    if(i==n-1),                     %1 次幂输出字符串'x'
        Y=[Y 'x'];
    elseif(i==n),                   %0 次幂不输出 x^n
        Y=Y;
    else
        Y=[Y 'x^' num2str(n-i)];    %其他情况输出 x^n
    end;
                                    %如果不是最后一项，输出'+'
end;
if(Y(1)=='+')               %修正如果 0 次幂为 0 时，造成字符串末尾有多余的字符串' + '
    Y(1)=[];
end;
```

以下语句说明了函数 poly2str()的用法：

```
poly2str([0 1 -2 3 -5 0 1 0 -1])
```

上述语句得到输出代码如下：

```
ans =
x^7-2x^6+3x^5-5x^4+x^2-1
```

上面字符串中 x^n 表示 x 的 n 次幂。

例如，设计一个函数实现从多项式的字符串表示转换为多项式的行向量表示，要求输入不必降幂排列，并且函数具有实现多项式同类项合并功能，代码设置如下：

```
%str2poly.m
%把多项式的字符串表示转换为行向量表示
function Y=str2poly(X)
%X 是字符串形式的多项式
%Y 是行向量形式的多项式
%输入格式检查
if(ischar(X)==0)
    disp('输入错误：输入 X 必须是一个字符串！');
end;
%用正则表达式寻找'+'或者'-'的下标位置
index=regexp(X,'\+|\-');
%多项式的项数
L=length(index);
%用于存储多项式每一项信息的单元字符串矩阵
term=cell(1,L+1);
term(1)=cellstr(X(1:(index(1)-1)));
for i=1:L-1,
    term(i+1)=cellstr(X(index(i):(index(i+1)-1)));
end;
term(L+1)=cellstr(X(index(L):end));
%如果第一项为空，则删除该项
if(isempty(char(term(1))))
    term(1)=[];
%多项式的项数减一
    L=L-1;
end;
%多项式系数矩阵
coefficient=[];
%多项式幂次矩阵，它与多项式系数矩阵一一对应
power=[];
for i=1:L+1,
%单项多项式字符串表达式
```

```
        substring=char(term(i));
%用正则表达式，寻找字符串'x^'，由于'^'是正则表达式中的特殊字符，多以用'\^'来表示
        index2=regexp(substring,'x\^');
        if(isempty(index2)==0),
%如果匹配上
            if(index2(1)==1),
%单项多项式字符串为'x^*'形式
                coefficient=[coefficient 1];
                power=[power str2num(substring((index2(1)+2):end))];
            elseif(index2(1)==2),
%单项多项式字符串为'+x^*'或者'-x^*'，'3x^*'形式
               if(substring(1)=='+'),
                   coefficient=[coefficient 1];
                   power=[power str2num(substring((index2(1)+2):end))];
               elseif(substring(1)=='-'),
                   coefficient=[coefficient -1];
                   power=[power str2num(substring((index2(1)+2):end))];
               end;
            else
%单项多项式字符串为'2.2x^*'
                coefficient=[coefficient str2num(substring(1:(index2(1)-1)))];
                power=[power str2num(substring((index2+2):end))];
            end;
        else
%单项多项式字符串不含'x^'
%用正则表达式，寻找字符串'x'
            index2=regexp(substring,'x');
            if(isempty(index2)==0),
%如果匹配上'x'
                if(index2(1)==1),
%单项多项式字符串为'x*'形式
                    coefficient=[coefficient 1];
                    power=[power 1];
                elseif(index2(1)==2),
%单项多项式字符串为'+x*'或者'-x*'，'3x*'形式
                   if((substring(1)=='+')==1),
                       coefficient=[coefficient 1];
                       power=[power 1];
                   elseif(substring(1)=='-'),
                       coefficient=[coefficient -1];
```

```
                        power=[power 1];
                    else
%单项多项式字符串为'2.2x*'
                        coefficient=[coefficient str2num(substring(1:(index2-1)))];
                         power=[power 1];
                    end;
                 else
                     coefficient=[coefficient str2num(substring(1:(index2-1)))];
                     power=[power 1];
                 end;
             else
%单项多项式字符串不含'x^','x',则断定它是常数项
                 coefficient=[coefficient str2num(substring)];
                 power=[power 0];
             end;
    end;
end;
%合并同类项
N=max(power)+1;
Y=zeros(1,N);
for i=1:N,
    index3=find(power==(N-i));
    Y(i)=sum(coefficient(index3));
end;
```

以下语句把一个多项式的字符串表示转换为一个多项式的向量表示，然后再把多项式的向量表示转回成多项式的字符串表示，代码设置如下：

```
X='-10x^2+8x^2-x^3+4x^5+x^3-10+9-x^7';
P=str2poly(X)
Str=poly2str(P)
```

由上述语句得到输出代码如下：

```
P =
    -1     0     4     0     0    -2     0    -1
Str =
-1x^7+4x^5-2x^2-1
```

由上述结果可以看出，输入多项式的同类项被合并了，这正是我们设计函数 str2poly()时所希望的。

6.1.2 多项式求值

在 MATLAB 7.0 中使用函数 polyval()来计算多项式的值。其调用方式如下：

- y = polyval(p,x)，p 为行向量形式的多项式，x 代入多项式的值，它可以是标量、向量和矩阵。如果 x 是向量或者矩阵，那么该函数将对向量或者矩阵的每一个元素计算多项式的值，并返回给 y。

MATLAB 7.0 不但可以计算矩阵元素的多项式值，也可以把整个矩阵代入多项式作为自变量进行计算。计算矩阵多项式值的函数是 polyvalm()，其调用方式如下：

- Y = polyvalm(p,X)，把矩阵 X 代入多项式 p 中进行计算，其中矩阵 X 必须是方阵。

例如，求多项式的值和矩阵多项式的值，请注意这两者的区别，具体代码如下：

```
%polyval_example.m
%多项式求值
str1='x^2+2x+3';
p1=str2poly(str1);
A=[1 0;
   0 -1];
p1_A=polyval(p1,A)                    %求多项式值
p1_Am=polyvalm(p1,A)                  %求矩阵多项式的值
```

由上述语句得到输出代码如下：

```
p1_A =
     6     3
     3     2
p1_Am =
     6     0
     0     2
```

6.1.3 多项式乘法和多项式除法

多项式乘法和除法运算也就是多项式向量的卷积和解卷积运算。在 MATLAB 7.0 中，函数 conv()和 deconv()用来实现这些运算。这两个函数的调用格式如下。

- w = conv(u,v)，实现向量 u,v 的卷积，在代数上相当于多项式 u 乘上多项式；
- [q,r] = deconv(v,u)，实现解卷积运算，他们之间的关系为 v = conv(u,q)+r。在代数上相当于实现多项式 u 除以 v，得到商 q 和余多项式 r。

例如，求多项式 x^2+2x+3 与多项式 $2x^2+3x+4$ 的乘积，代码设置如下：

```
%poly_conv.m
%多项式乘法
str1='x^2+2x+3';
str2='2x^2+3x+4';
p1=str2poly(str1);
p2=str2poly(str2);
p3=conv(p1,p2);
str3=poly2str(p3)
```

上述语句得到输出如下：

```
str3 =
2x^4+7x^3+16x^2+17x+12
```

为了直观，上面程序利用了 6.1.1 小节中的多项式的字符串表示与多项式的向量表示的互换函数。

例如，求上个例子中的多项式乘积结果 $2x^4+7x^3+16x^2+17x+12$ 除以多项式 x^2+2x+3 的商，代码设置如下：

```
%poly_deconv.m
%多项式除法
str3='2x^4+7x^3+16x^2+17x+12';
str1='x^2+2x+3';
p3=str2poly(str3);
p1=str2poly(str1);
[p2 r]=deconv(p3,p1);
str2=poly2str(p2)
r
```

由上述语句得到输出代码如下：

```
str2 =
2x^2+3x+4
r =
     0     0     0     0     0
```

6.1.4　多项式的导数和微分

1．多项式的导数

在 MATLAB 7.0 中使用函数 polyder()来计算多项式的导数，其调用方式如下：

- k = polyder(p)，返回多项式 p 的导数；
- k = polyder(a,b)，返回多项式 a 与多项式 b 乘积的导数；
- [q,d] = polyder(b,a)，返回多项式 a 除以 b 的商的导数，并以 q/d 格式表示。

例如，对多项式求导，代码设置如下：

```
%polyder_example.m
%多项式求导数的示例
str1='x^2+2x+3';
str2='2x^2+3x+4';
p1=str2poly(str1);
p2=str2poly(str2);
%求多项式'x^2+2x+3'的导数
q1=polyder(p1);
str1_der=poly2str(q1)
%求多项式'2x^2+3x+4'的导数
```

```
q2=polyder(p2);
str2_der=poly2str(q2)
%求多项式'x^2+2x+3'与'2x^2+3x+4'乘积的导数
c=polyder(p1,p2);
str3_der=poly2str(c)
%求多项式'x^2+2x+3'除以'2x^2+3x+4'的商的导数
[q d]=polyder(p1,p2);
str_der_q=poly2str(q)
str_der_q=poly2str(d)
```

由上述语句得到输出代码如下：

```
str1_der =
2x+2
str2_der =
4x+3
str3_der =
8x^3+21x^2+32x+17
str_der_q =
-1x^2-4x-1
str_der_q =
4x^4+12x^3+25x^2+24x+16
```

2．多项式的积分

在 MATLAB 7.0 中使用函数 polyint()来计算多项式的积分，其调用方式如下：

- polyint(p,k)，返回多项式 p 的积分，设积分的常数项为 k；
- polyint(p)，返回多项式 p 的积分，设积分的常数项为 0。

例如，求多项式 $3x^2+4x+5$ 的积分，代码设置如下：

```
%polyint_example.m
%多项式求积分的示例
str1='3x^2+4x+5';
p1=str2poly(str1);
%求积分并设常数项为 5
p2=polyint(p1,5);
str2=poly2str(p2)
%求积分并设常数项为 0
p3=polyint(p1);
str3=poly2str(p3)
```

由上述语句得到输出代码如下：

```
str2 =
x^3+2x^2+5x+5
str3 =
```

```
x^3+2x^2+5x
```

6.1.5 多项式的根和由根创建多项式

1．多项式的根

在 MATLAB 7.0 中使用函数 roots()来求多项式的根，其调用方式如下：

- *r* = roots(*c*)，返回多项式 *c* 的所有根 *r*，*r* 是向量，其长度等于根的个数。

例如，求多项式 $2x^3-2x^2-8x+8$ 的根，代码设置如下：

```
%roots_example.m
%多项式求根
str1='2x^3-2x^2-8x+8';
p1=str2poly(str1);
r=roots(p1)                          %求多项式根
```

由上述语句得到输出代码如下：

```
r =
    -2.0000
     2.0000
     1.0000
```

2．由根创建多项式

与多项式求根相反的过程是由根创建多项式，它由函数 poly()来实现，其调用格式如下：

- *p* = poly(*r*)，输入 *r* 是多项式所有根，返回值为代表多项式的行向量形式；
- *p* = poly(*A*)，输入是 $N\times N$ 的方阵，返回值 *p* 是长度为 $N+1$ 的行向量多项式，它是矩阵 ***A*** 的特征多项式，也就是说多项式 *p* 的根是矩阵 *A* 的特征值。

例如，由根[-2 2 1]创建多项式，代码设置如下：

```
%poly_example.m
%由根创建多项式
r=[-2 2 1];
p=poly(r);
str=poly2str(p)
```

由上述语句得到输出代码如下：

```
str =
x^3-1x^2-4x+4
```

例如，求矩阵的特征多项式，代码设置如下：

```
%poly_example2.m
%求矩阵 A 的特征多项式
A =[1     2     3;
    4     5     6;
    7     8     0];
```

```
p=poly(A);
str=poly2str(p)
r=roots(p)                %求多项式 p 的根
A_eig=eig(A)              %求矩阵 A 的特征值
```

由上述语句得到输出代码如下：

```
str =
x^3-6x^2-72x-27
r =
    12.1229
    -5.7345
    -0.3884
A_eig =
    12.1229
    -0.3884
    -5.7345
```

6.1.6　多项式部分分式展开

函数 residue()可以将多项式之比用部分分式展开，也可以将一个部分分式表示为多项式之比。函数 residue()的调用格式如下：

- $[r,p,k]$ = residue(b,a)，求多项式之比 b/a 的部分分式展开，返回值 r 是部分分式的留数，p 是部分分式的极点，k 是直接项。如果多项式 a 没有重根，部分分式的形式如下：

$$\frac{b(x)}{a(x)}=\frac{r_1}{x-p_1}+\frac{r_2}{x-p_2}+\dots+\frac{r_n}{x-p_n}+k_s$$

其中向量 r，p 的长度和向量 a,b 的长度有如下关系：

$$\text{length}(a)-1=\text{length}(r)=\text{length}(p)$$

当向量 b 的长度小于 a 时，向量 k 中没有元素，否则应满足如下关系：

$$\text{length}(k)=\text{length}(b)-\text{length}(a)+1$$

- $[b,a]$ = residue(r,p,k)，从部分分式得到多项式向量。

例如，求多项式之比的部分分式展开，然后在把部分分式表示为多项式之比的形式，代码设置如下：

```
%residue_example.m
%求多项式之比的部分分式，然后把部分分式表示为多项式之比的形式
str1='2x^3+3x^2-4x+1';
str2='x^2-3x+2';
p1=str2poly(str1);
p2=str2poly(str2);
[r,p,k]=residue(p1,p2)                %求多项式之比的部分分式
```

```
[p1_res,p2_res]=residue(r,p,k);                %把部分分式表示为多项式之比的形式
str1_res=poly2str(p1_res)                      %把多项式显示为字符串形式
str2_res=poly2str(p2_res)
```

由上述语句得到输出代码如下：

```
r =
    21
    -2
p =
     2
     1
k =
     2     9
str1_res =
2x^3+3x^2-4x+1
str2_res =
x^2-3x+2
```

6.1.7 多项式曲线拟合

函数 polyfit()采用最小二乘法对给定数据进行多项式拟合，最后给出多项式的系数。该函数的调用方式如下：

- *p* = polyfit(*x*,*y*,*n*)，采用 *n* 次多项式 *p* 来拟合数据 *x* 和 *y*，从而使得 *p(x)*与 *y* 最小均方差最小。

例如，下面例子说明函数 polyfit()的用法，并讨论采用不同多项式阶数对拟合结果的影响，代码设置如下：

```
%polyfit_example.m
%说明函数 polyfit()的用法，并讨论采用不同多项式阶数对拟合结果的影响
x=0:0.2:10;
y=0.25*x+20*sin(x);
%5 阶多项式拟合
p5=polyfit(x,y,5);
y5=polyval(p5,x);
%8 阶多项式拟合
p8=polyfit(x,y,8);
y8=polyval(p8,x);
%60 阶多项式拟合
p60=polyfit(x,y,60);
y60=polyval(p60,x);
%画图
hold on;
```

```
plot(x,y,'ro');
plot(x,y5,'b--');
plot(x,y8,'b:');
plot(x,y60,'r-.');
xlabel('x');
ylabel('y');
legend('原始数据','5 阶多项式拟合','8 阶多项式拟合','60 阶多项式拟合');
```

得到输出如图 6-1 所示。由图 6-1 可以看出：使用 5 次多项式拟合时，拟合得到的结果比较差。而使用 8 次多项式拟合时，得到的结果与原始数据符合得很好。但使用 60 次多项式拟合时，拟合的结果非常差。可见用多项式拟合必须选择适中的阶数，而不是阶数越高精度越高。

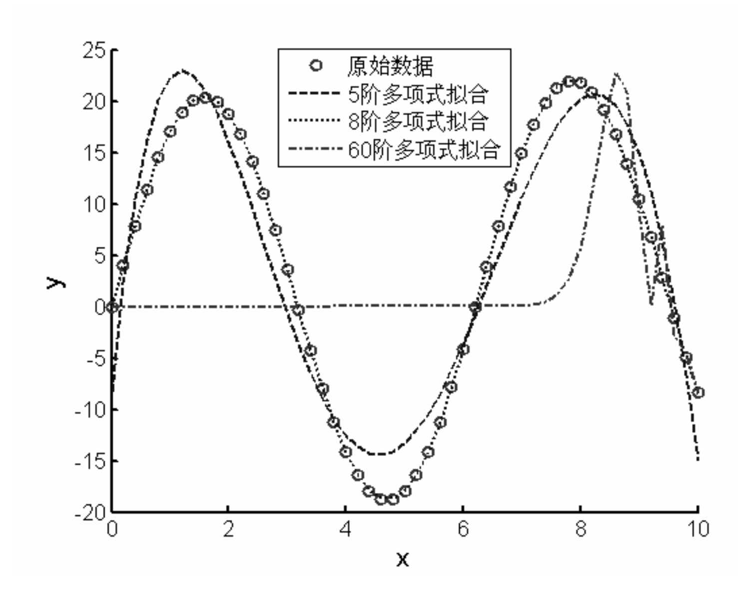


图 6-1　多项式曲线拟合

6.1.8　曲线拟合图形用户接口

为了方便用户的使用，MATLAB 7.0 提供了支持曲线拟合的图形用户接口。它位于“Figure”窗口的“Tools\Basic Fitting”菜单中。为了使用该工具，首先用待拟合的数据画图，代码设置如下：

```
x=0:0.2:10;
y=0.25*x+20*sin(x);
plot(x,y,'ro');
```

得到“Figure”窗口，如图 6-2 所示。然后在“Figure”窗口中点击菜单“Tools\Basic Fitting”，得到“Basic Fitting”窗口，如图 6-3 所示。可以单击“Basic Fitting”窗口右下角的向右按钮，得到“Basic Fitting”窗口的全貌，如图 6-4 所示。

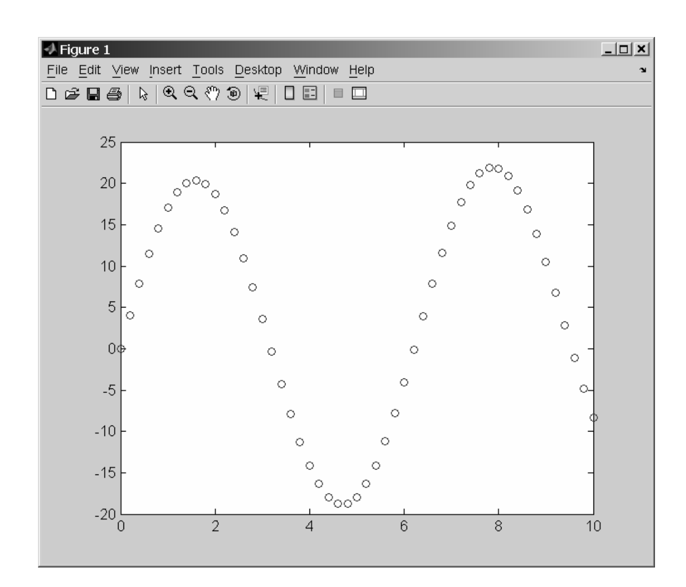


图 6-2 “Figure”窗口

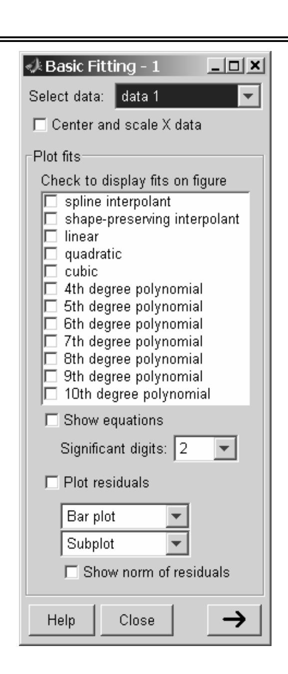


图 6-3 Basic Fitting 窗口

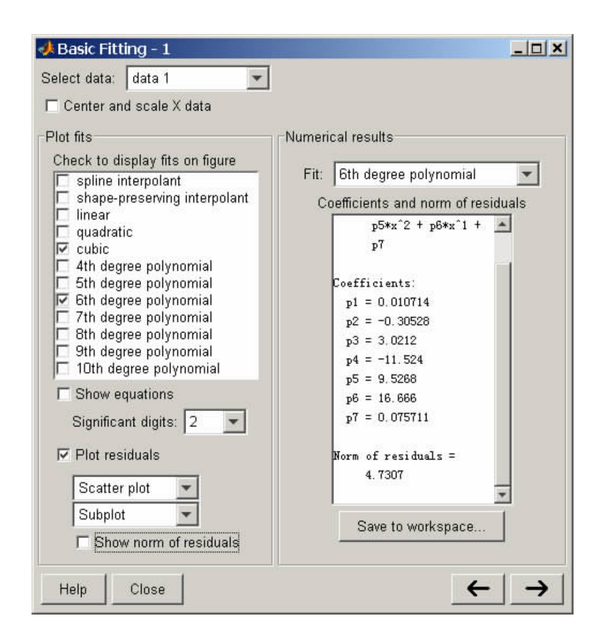


图 6-4 “Basic Fitting”窗口全貌

在“Basic Fitting”窗口的“Plot fits”复选框中选择“cubic”和“6th degree polynomial”这两选项，然后单击“Plot residuals”单选框使之处在选上状况，最后把“Plot residuals”单选框正下方的下拉列表选择为“Scatter plot”。在这些操作后，“Figure”窗口已经出现拟合好的数据，并且画出拟合数据的残留误差 residuals（如图 6-5 所示）。同时拟合结果也在“Basic Fitting”的“Numerical results”中显示出来。

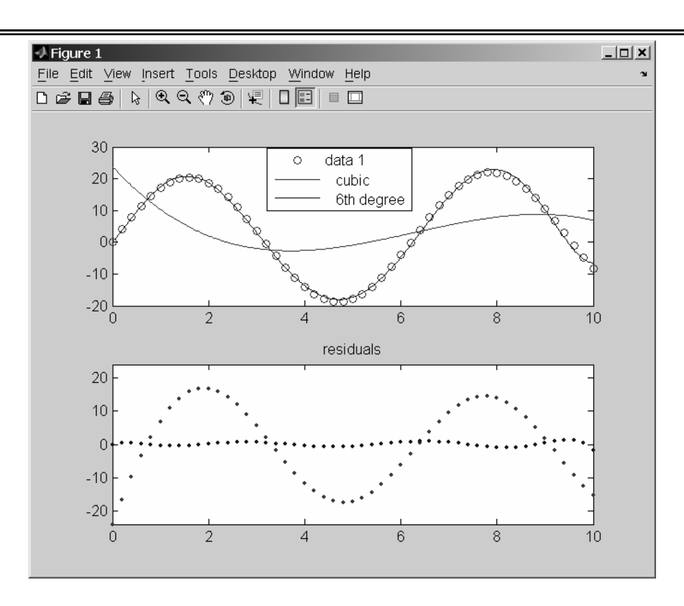


图 6-5　数据拟合结果

6.2　插值

插值是在已知数据之间寻找估计值的过程。在信号处理和图像处理中，插值是极其常用的方法。MATLAB 7.0 提供大量的插值函数。这些函数在获得数据的平滑度、时间复杂度和空间复杂度方面上有不同的性能。

插值函数保存在 MATLAB 7.0 工具箱的 polyfun 子目录下，如表 6-2 所示。

表 6-2　插值函数

函数名	功能描述
pchip	分段三次厄米多项式插值
interp1	一维插值
interp1q	一维快速插值
interpft	一维快速傅立叶插值方法
interp2	二维插值
interp3	三维插值
interpn	N 维插值
griddata	栅格数据插值
griddata3	3 维栅格数据插值
griddatan	N 维栅格数据插值
spline	三次样条插值
ppval	分段多项式求值

6.2.1　一维插值

一位插值就是对一维函数 $y=f(x)$进行插值，图 6-6 显示了插值的含义，实心点(x,y)代表的是已知数据，空心点(xi,yi)的横坐标 xi 代表需要估计值的位置，纵坐标 yi 表示插值后的估计

值。在 MATLAB 7.0 中，一维插值有基于多项式的插值和基于快速傅立叶的插值两种类型。

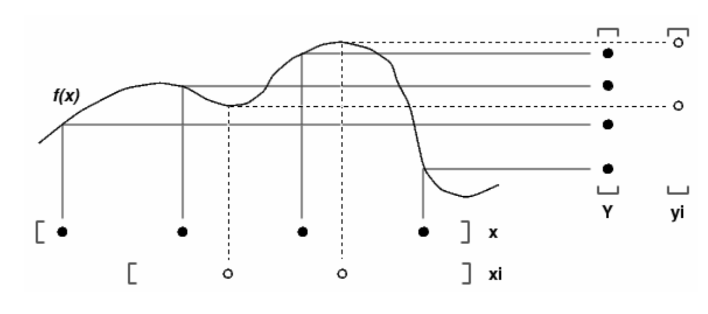


图 6-6　一维插值示意图

1．一维多项式插值

一维多项式插值可以通过函数 interp1()来实现。函数 interp1()的基本调用格式如下。

- *yi*=interp1(*x*,*y*,*xi*,*method*)，*x* 必须是向量，*y* 可以是向量也可以是矩阵。如果 *y* 是向量，则必须与 *x* 具有相同的长度，这时 *xi* 可以是标量、向量和任意维矩阵，*yi* 与 *xi* 具有相同的大小；如果 *y* 是矩阵，则其大小必须是[*n*,*d*1,*d*2,...,*dk*]（*n* 为向量 *x* 的长度），函数对 *d*1**d*2*... **dk* 组 *y* 值都进行插值；
- *yi* = interp1(*y*,*xi*)，默认 *x* 为 1:*n*，其中 *n* 为向量 *y* 的长度；
- *yi* = interp1(*x*,*y*,*xi*,*method*)，输入变量 *method* 用于指定插值的方法。可采用的插值方法将在本小节详细介绍；
- *yi* = interp1(*x*,*y*,*xi*,*method*,'extrap')，对超出数据范围的插值数据指定外推方法'extrap'；
- *yi* = interp1(*x*,*y*,*xi*,*method*,*extrapval*)，对超出数据范围的插值数据返回 *extrapval* 值，一般设为 NaN 或者 0；
- *pp* = interp1(*x*,*y*,*method*,'pp')，返回值 *pp* 为数据 *y* 的分段多项式形式。*method* 指定产生分段多项式 *pp* 的方法，它可以是插值方法中除了'v5cubic'的任何方法。

一维插值可以指定的方法如下：

（1）最邻近插值（Nearest neighbor interpolation，method='nearest'），这种插值方法在已知数据的最邻近点设置插值点，对插值点的数进行四舍五入。对超出范围的点将返回一个 NaN（Not a Number）。

（2）线性插值（Linear interpolation，method='linear'），该方法是未指定插值方法时 MATLAB7.0 默认采用的方法。该方法采用直线连接相邻的两点。对超出范围的点将返回一个 NaN（Not a Number）。

（3）三次样条插值（Cubic spline interpolation，method='spline'），这样方法采用三次样条函数来获得插值点。在已知点为端点情况下，插值函数至少具有相同的一阶和二阶导数。

（4）分段三次厄米多项式插值（Piecewise cubic Hermite interpolation，method='pchip'）。

（5）三次多项式插值（method='cubic'），与分段三次厄米多项式插值相同。

（6）MATLAB5 中使用的三次多项式插值（method='v5cubic'），该方法使用一个 3 次多项式函数对已知数据进行拟合。

当选择一个插值方法时应该考虑方法的执行速度、占用内存大小和获得数据的平滑度。以上方法的特点如下。

（1）最邻近插值：最快的插值方法，但是数据平滑方面最差，其得到的数据是不连续的。

（2）线性插值：比邻近插值占用更多的内存，执行速度也稍慢，但其数据平滑方面优于邻近插值。与邻近插值不同的是，线性插值的数据变化是连续的。

（3）三次样条插值：处理速度最慢，占用内存小于分段三次厄米多项式插值，可以产生最光滑的结果。但是如果输入数据不均匀或者某些点靠得很近，会出现一些错误。样条插值是极其有用的插值方法，因此除了提供三次样条插值函数外，MATLAB 7.0 还提供了一个样条插值工具箱，它位于 toolbox\splines 下。

（4）分段三次厄米多项式插值：处理速度和消耗的内存比线性插值差，但插值得到的数据和一阶导数都是连续的。

这些方法的相对优劣不仅适用于一维插值，而且适用于二维插值情况甚至高维插值情况。

例如，下面例子用不同插值方法对一维数据进行插值，并比较其不同，代码设置如下：

```
%interp1_example.m
%用不同插值方法对一维数据进行插值，并比较其不同
x = 0:1.2:10;
y = sin(x);
xi = 0:0.1:10;
yi_nearest = interp1(x,y,xi,'nearset'); %最邻近插值
yi_linear = interp1(x,y,xi);                    %默认插值方法是线性插值
yi_spline = interp1(x,y,xi,'spline ');    %三次样条插值
yi_cubic = interp1(x,y,xi,'cubic');         %三次多项式插值
yi_v5cubic = interp1(x,y,xi,'v5cubic'); %MATLAB 5 中使用的三次多项式插值
hold on;
subplot(2,3,1);
plot(x,y,'ro',xi,yi_nearest,'b-');
title('最邻近插值');
subplot(2,3,2);
plot(x,y,'ro',xi,yi_linear,'b-');
title('线性插值');
subplot(2,3,3);
plot(x,y,'ro',xi,yi_spline,'b-');
title('三次样条插值');
subplot(2,3,4);
plot(x,y,'ro',xi,yi_cubic,'b-');
title('三次多项式插值');
subplot(2,3,5);
plot(x,y,'ro',xi,yi_v5cubic,'b-');
title('三次多项式插值(MATLAB5)');
```

由上述语句得到如图 6-7 所示的示意图。

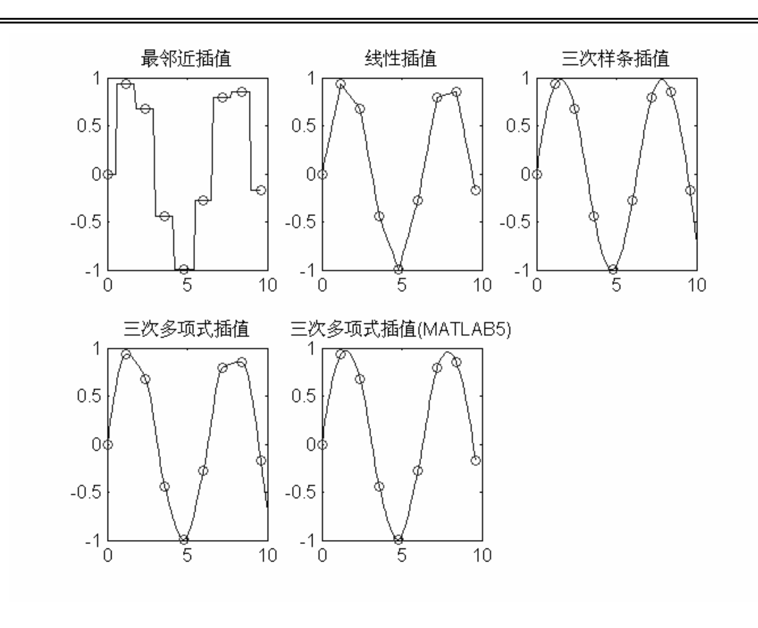


图 6-7　一维插值

2．一维快速傅立叶插值

一维快速傅立叶插值通过函数 interpft()来实现，该函数用傅立叶变换把输入数据变换到频域，然后用更多点的傅立叶逆变换，变换回时域，其结果是对数据进行增采样。函数 interpft()的调用格式如下。

- *y* = interpft(*x*,*n*)，对 *x* 进行傅立叶变换，然后采用 *n* 点傅立叶逆变换变回到时域。如果 *x* 是一个向量，数据 *x* 的长度为 *m*，采样间隔为 *dx*，则数据 *y* 的采样间隔是 *dx***m*/*n*，注意 *n* 值必须大于 *m*；如果 *x* 是矩阵，函数操作在 *x* 的列上，返回结果与 *x* 具有相同的列数但其行数为 *n*；
- *y* = interpft(*x*,*n*,*dim*)，在 *dim* 指定的维度上进行操作。

例如，利用一维快速傅立叶插值实现数据增采样，代码设置如下：

```
%interpft_example.m
%一维快速傅立叶插值实现数据增采样
x = 0:1.2:10;
y = sin(x);
n = 2*length(x);                %增采样 1 倍
yi = interpft(y,n);             %一维快速傅立叶插值
xi = 0:0.6:10.4;
hold on;
plot(x,y,'ro');                 %画图
plot(xi,yi,'b.-');
title('一维快速傅立叶插值');
legend('原始数据','插值结果');
```

由上述语句得到如图 6-8 所示的示意图。

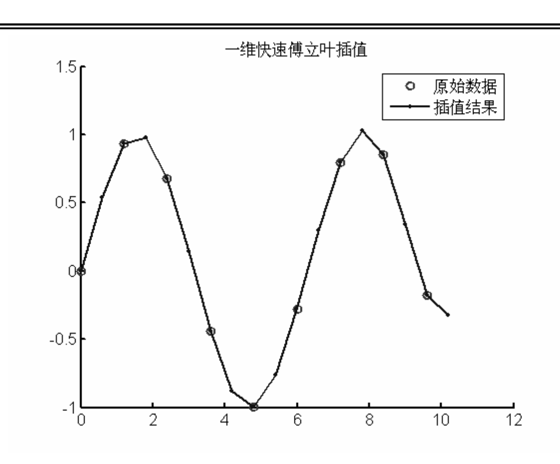


图 6-8　一维快速傅立叶插值

6.2.2　二维插值

二维插值主要应用于图像处理和数据的可视化，其基本思想与一维插值相同，它是对两变量的函数 z=f(x,y)进行插值。二维插值的示意图如图 6-9 所示。

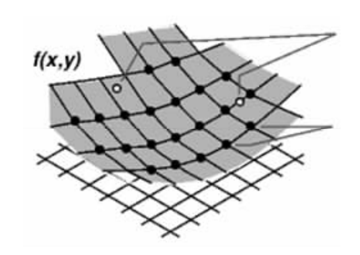


图 6-9　二维插值示意图

MATLAB 7.0 中的二维插值函数为 interp2()，其调用格式如下。

- *zi* = interp2(*x*,*y*,*z*,*xi*,*y*i)，原始数据 *x*,*y*,*z* 决定插值函数 *z*=*f*(*x*,*y*)，返回值 *zi* 是(*xi*,*yi*)在函数 f(*x*,*y*)上的值；
- *zi* = interp2(*z*,*xi*,*yi*)，若 *z*=*n*×*m*，则 *x*=*1*:*n*，*y*=*1*:*m*；
- *zi* = interp2(*z*,*ntimes*)，在两点之间递归地插值 *ntimes* 次；
- *zi* = interp2(*x*,*y*,*z*,*xi*,*yi*,*method*)，可采用的插值方法在本小节将详细介绍；
- *zi* = interp2(...,*method*, *extrapval*)，当数据超过原始数据范围时，用输入 *extrapval* 指定一种外推方法。

二维插值可以采用的插值方法如下：

（1）最邻近插值（Nearest neighbor interpolation，method='nearest'），这种插值方法在已知数据的最邻近点设置插值点，对插值点的数进行四舍五入。对超出范围的点将返回一个 NaN（Not a Number）。

（2）双线性插值（Bilinear interpolation，method='linear'）,该方法是未指定插值方法时 MATLAB7.0 默认采用的方法。插值点的值只决定于最邻近的 4 个点的值。

（3）三次样条插值（Cubic spline interpolation，method='spline'），这样方法采用三次样条

函数来获得插值数据。

（4）双三次多项式插值（method='cubic'）。

例如，下面例子比较各种二维插值插值算法的不同，如下：

```
%interp2_example2
%采用二次插值对三维高斯型分布函数进行插值
[x,y] = meshgrid(-3:0.8:3);                %原始数据
z = peaks(X,Y);
[xi,yi] = meshgrid(-3:0.25:3);             %插值点
zi_nearest = interp2(x,y,z,xi,yi,'nearset');   %最邻近插值
zi_linear = interp2(x,y,z,xi,yi);          %默认插值方法是线性插值
zi_spline = interp2(x,y,z,xi,yi,'spline ');    %三次样条插值
zi_cubic = interp2(x,y,z,xi,yi,'cubic');   %三次多项式插值
hold on;
subplot(2,3,1);
surf(x,y,z);
title('原始数据');
subplot(2,3,2);
surf(xi,yi,zi_nearest);
title('最邻近插值');
subplot(2,3,3);
surf(xi,yi,zi_linear);
title('线性插值');
subplot(2,3,4);
surf(xi,yi,zi_spline);
title('三次样条插值');
subplot(2,3,5);
surf(xi,yi,zi_cubic);
title('三次多项式插值');
figure;                                    %新开绘图窗口
subplot(2,2,1);                            %画插值结果的等高线
contour(xi,yi,zi_nearest);
title('最邻近插值');
subplot(2,2,2);
contour(xi,yi,zi_linear);
title('线性插值');
subplot(2,2,3);
contour(xi,yi,zi_spline);
title('三次样条插值');
subplot(2,2,4);
contour(xi,yi,zi_cubic);
```

```
title('三次多项式插值');
```

由上述代码得到的效果图如图 6-10 和图 6-11 所示。如图 6-11 所示为插值后数据的等高线，可以用于比较插值后数据的平滑性能。

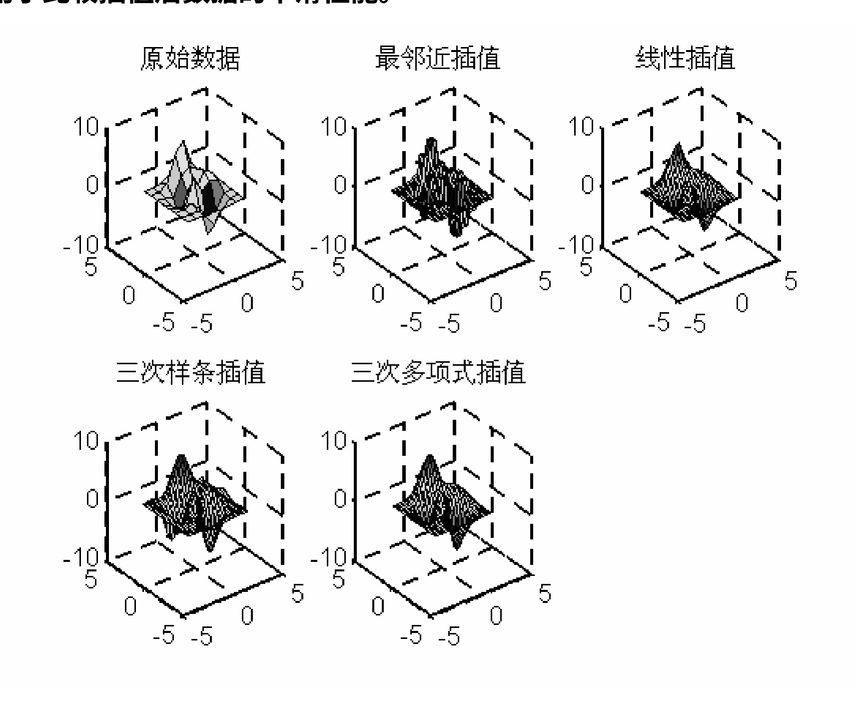


图 6-10　二维插值

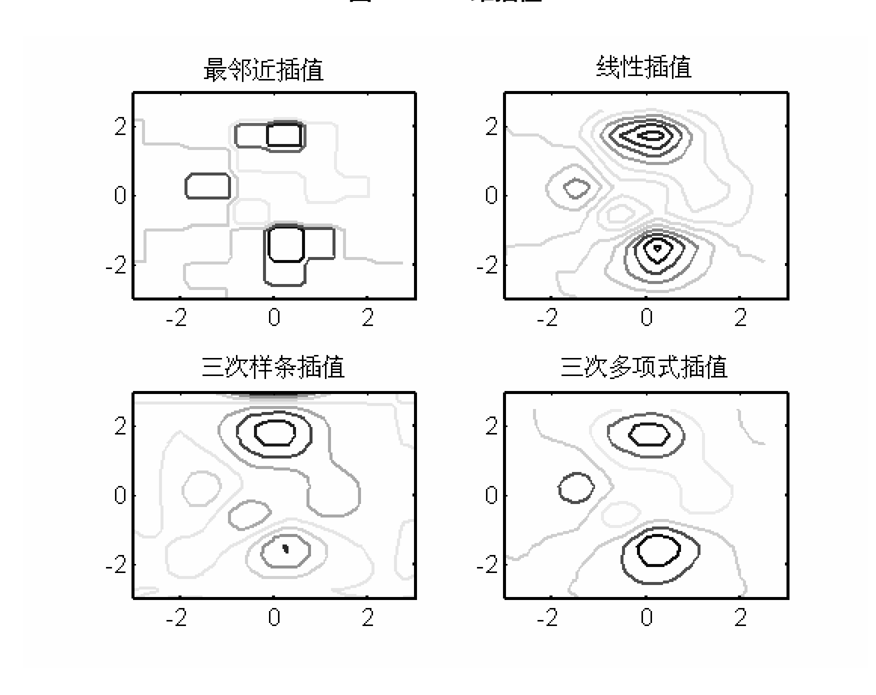


图 6-11　二维插值后的等高线

6.3 数据分析和傅立叶变换

MATLAB7.0 中与数据分析和傅立叶变换相关的函数位于目录\toolboxs\MATLAB\datafun下，本节将分类介绍这些函数。

MATLAB 7.0 在这些函数中的约定。

- 一维统计数据可以用行向量或者列向量来表示，不管输入数据是行向量或者是列向量，运算是对整个向量进行的；
- 二维统计数据可以采用多个向量表示，也可以采用二维矩阵来表示。对二维矩阵运算时，运算总是按列进行的。

这两条约定不仅适用于本节提到的函数，而且适用于 MATLAB 7.0 各个工具箱的函数。

6.3.1 基本数据分析函数

MATLAB 7.0 提供的基本数据分析函数的功能和调用格式如表 6-3 所示。

表 6-3　　基本数据分析函数

函数名	功能描述	基本调用格式
max	求最大值	*C* = max(*A*)，如果 *A* 是向量，返回向量中的最大值；如果 *A* 是矩阵返回一个包含各列最大值的行向量 *C* = max(*A*,*B*)，返回矩阵 *A* 和 *B* 之中较大的元素，矩阵 *A*、*B* 必须具有相同的大小 *C* = max(*A*,[],*dim*)，返回 *dim* 维上的最大值 [*C*,*I*] = max(...)，多返回最大值的下标
min	求最小值	与最大值函数 max()调用格式一致
mean	求平均值	*M* = mean(*A*)，如果 *A* 是向量，返回向量 *A* 的平均值，如果 *A* 是矩阵，返回含有各列平均值的行向量 *M* = mean(*A*,*dim*)，返回 *dim* 维上的平均值
median	求中间值	与求平均值函数 mean()的调用格式一致
std	求标准方差	*s* = std(*A*)，如果 *A* 是向量，返回向量的标准方差；如果 *A* 是矩阵，返回含有各列标准方差的行向量 *s* = std(*A*,*flag*)，用 *flag* 选择标准方差的定义式 *s* = std(*A*,*flag*,*dim*)，返回 *dim* 维上的标准方差
var	方差（标准方差的平反）	var(*X*)，如果 *A* 是向量，返回向量的方差；如果 *A* 是矩阵，返回含有各列方差的行向量 var(*X*,1)，返回第二种定义的方差 var(*X*,*w*)，利用 *w* 做为权重计算反差 var(*X*,*w*,*dim*)，返回 *dim* 维上的方差
sort	数据排序	*B* = sort(*A*)，如果 *A* 是向量，升序排列向量；如果 *A* 是矩阵，升序排列各个列 *B* = sort(*A*,*dim*)，升序排列矩阵 *A* 的 *dim* 维 *B* = sort(...,*mode*)，用 *mode* 选择排序方式：'ascend'为升序，'descend'为降序 [*B*,*IX*] = sort(...)，多返回数据 *B* 在原来矩阵中的下标 *IX*
sortrows	对矩阵的行排序	*B* = sortrows(*A*)，升序排序矩阵 *A* 的行 *B* = sortrows(*A*,*column*)，以 *column* 列数据作为标准，升序排序矩阵 *A* 的行 [*B*,*index*] = sortrows(*A*)，多返回数据 *B* 在原来矩阵 *A* 中的下标 *IX*

续表

sum	求元素之和	*B* = sum(*A*)，如果 *A* 是向量，返回向量 *A* 的各元素之和；如果 *A* 是矩阵，返回含有各列元素之和的行向量 *B* = sum(*A*, *dim*)，求 *dim* 维上的矩阵元素之和 *B* = sum(*A*, 'double')，返回数据类型指定为双精度浮点数 *B* = sum(*A*, 'native')，返回数据类型指定为与矩阵 *A* 的数据类型相同
prod	求元素的连乘积	*B* = prod(*A*)，如果 *A* 是向量，返回向量 *A* 的各元素连乘积；如果 *A* 是矩阵，返回含有各列元素连乘积的行向量 *B* = prod(*A*,*dim*)，返回 *dim* 维上的矩阵元素连乘积
hist	画直方图	*n* = hist(*Y*)，在 10 个等间距的区间统计矩阵 *Y* 属于该区间的元素个数 *n* = hist(*Y*,*x*)，在 *x* 指定的区间统计矩阵 *Y* 属于该区间的元素个数 *n* = hist(*Y*,*nbins*),用 *nbins* 个等间距的区间统计矩阵 *Y* 属于该区间的元素个数 hist(...)，直接画出直方图
histc	直方图统计	*n* = histc(*x*,*edges*)，计算在 *edges* 区间内向量 x 属于该区间的元素个数 *n* = histc(*x*,*edges*,*dim*)，在 *dim* 维上统计 *x* 出现的次数
trapz	梯形数值积分（等间距）	*Z* = trapz(*Y*)，返回 *Y* 的梯形数值积分 *Z* = trapz(*X*,*Y*)，计算以 *X* 为自变量时 *Y* 的梯形数值积分 *Z* = trapz(...,*dim*)，在 *dim* 维上计算梯形数值积分
cumsum	矩阵的累加	*B* = cumsum(*A*)，如果 *A* 是向量，计算向量 *A* 的累计和；如果 *A* 是矩阵，计算矩阵 *A* 在列方向上的累计和 *B* = cumsum(*A*,*dim*)，在 *dim* 维上计算矩阵 *A* 的累计和
cumprod	矩阵的累积	函数调用格式与函数 cumsum()相同
cumtrapz	梯形积分累计	函数调用格式与函数 trapz()相同

1．最大值、最小值、平均值、中间值、元素求和

这几个函数的用法都比较简单，下面的例子用来说明这些函数的用法：

```
%max_example
%最大值、最小值、平均值、中间值、元素求和函数使用的例子
x=1:40;
y=randn(1,40);                      %正态分布随机数据
hold on;
plot(x,y);
[y_max,I_max]=max(y);               %求向量最大值和相应下标
plot(x(I_max),y_max,'o');
[y_min,I_min]=min(y);               %求向量最小值和相应下标
plot(x(I_min),y_min,'*');
y_mean=mean(y);                     %求向量平均值
plot(x,y_mean*ones(1,length(x)),':');
y_median=median(y);                 %求向量反差
plot(x,y_median*ones(1,length(x)),'-.');
y_sum=sum(y);                       %求向量元素之和
plot(x,y_sum*ones(1,length(x)),'--');
```

```
xlabel('x');
ylabel('y');
legend('数据','最大值','最小值','平均值','中间值','向量元素之和');
```

由上述语句得到的效果如图 6-12 所示。

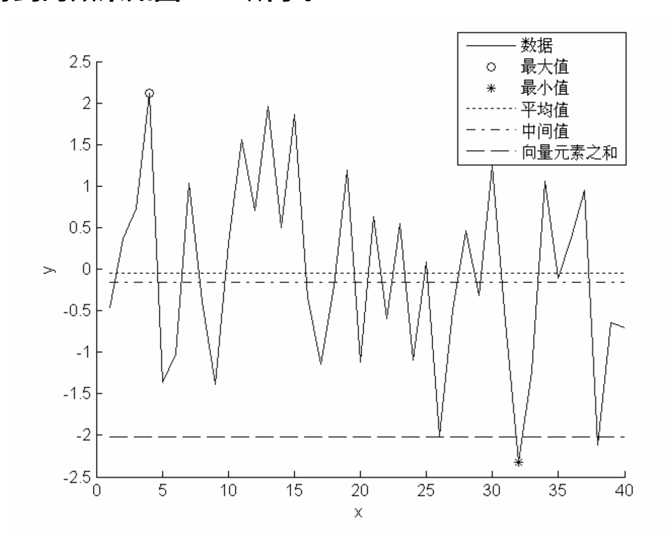


图 6-12　最大值、最小值、平均值、中间值、元素求和

2．标准方差和方差

向量 x 的标准方差有如下两种定义：

$$s=\left[\frac{1}{N-1}\sum_{k=1}^{N}(x_k-\overline{x})^2\right]^{\frac{1}{2}} \tag{5-1}$$

$$s=\left[\frac{1}{N}\sum_{k=1}^{N}(x_k-\overline{x})^2\right]^{\frac{1}{2}} \tag{5-2}$$

其中 $\overline{x}=\sum_{k=1}^{N}x_k$ ，N 是向量 x 的长度。

MATLAB 7.0 中默认使用（5-1）式来计算数据的标准方差。要使用（5-2）式来计算标准方差可以使用函数调用格式 std(A,1)。

方差是标准方差的平反。对应标准方差的两种定义，方差也有两种定义。同样 MATLAB 7.0 中默认使用（5-1）式来计算数据的方差。要使用（5-2）式来计算方差可以使用函数调用格式 var(*A*,1)。

例如，有一维随机变量 x，要求验证 $2*x$ 的方差是 x 的 4 倍，$2x$ 标准方差是 x 的 2 倍，代码设置如下：

```
%std_example.m
%验证 2*x 的方差是 x 的 4 倍，2x 标准方差是 x 的 2 倍
```

```
x=rand(1,1000);
x_std=std(x);
x_var=var(x);
x2_std=std(2*x);
x2_var=var(2*x);
disp(['随机变量 x2 的标准方差与 x1 的标准方差之比 = ' num2str(x2_std/x_std)]);
disp(['随机变量 x2 的标准方差与 x1 的标准方差之比 = ' num2str(x2_var/x_var)]);
```

由上述语句得到输出代码如下：

```
随机变量 x2 的标准方差与 x1 的标准方差之比 = 2
随机变量 x2 的标准方差与 x1 的标准方差之比 = 4
```

3．元素排序

MATLAB 7.0 可以对实数、复数和字符串进行排序。对复数矩阵进行排序时，先按复数的模进行排序，如果模相等则按其在区间[-π, π]上的相角进行排序。MATLAB7.0 中实现排序的函数为 sort()。

例如，对复数矩阵排列，代码设置如下：

```
%sort_complex.m
%对复数进行排序
a = [0 -1 -0.5i 1 0.5 i -i -0.5 0.5i];
b = sort(a)
```

由上述语句得到输出代码如下：

```
b =
     0    -0.5000i    0.5000      0.5000i      -0.5000    1.0000i    1.0000    1.0000i    -1.0000
```

MATLAB 7.0 提供函数 sortrows()用于对矩阵的行进行排列。

例如，对字符数据进行排列，代码设置如下：

```
%sortrows_string.m
%对字符数组进行排序
a = ['hello';'world';'hally';'Clayt';'Daney'];
b = sortrows(a)                              %按第 1 列的字符来排序字符串
c = sortrows(a,2)                            %按第 2 列的字符来排序字符串
```

由上述语句得到输出代码如下：

```
b =
Clayt
Daney
hally
hello
world
c =
hally
Daney
```

```
hello
Clayt
world
```

4．基本数据分析实例

例如，需要设计一个函数从已知 *n* 个数据中挑选出方差最小的 *k* 个数据。显然，如果要穷举所有的情况，计算量将会很大。本例中采用的算法是先对数据进行排队，然后计算所有相邻的 *k* 个数据的方差，选出方差最小的相邻 *k* 个数据就是所求的数据。本例在排序时记录了排序后数据在未排序数据中的下标，故能同时返回方差最小的 *k* 个数据在原来数据中的位置。从已知 *n* 个数据中挑选出方差最小的 *k* 个数据的函数如下：

```
%minvar.m
%x 是一个向量，k 是小于等于向量 x 长度的正标量
%返回值是 y 是 x 中反差最小的 k 个数据，i 是 y 的元素在 x 向量中的下标
function [y,i]=minvar(x,k)
if(isvector(x)==0),                          %输入数据检查
    disp('输入数据 x 必须是向量!');
    return;
end;
if(isscalar(k)==0)
    disp('输入数据 k 必须是标量!');
    return;
end;
if(k<=0||k>length(x)),
    disp('输入数据 k 必须是是小于等于向量 x 长度的正标量!');
    return;
end;
k=floor(k);                                  %把 k 截断为整数
[x_sort index]=sort(x);                      %对数据 x 排序
n=length(x);                                 %数据 x 的长度
for i=1:(n-k+1),                             %对排好序的数据求所有相邻 k 个值
    std_x_sort(i)=std(x_sort(i:(i+k-1)));    %的标准方差
end;
[min_std_x_sort j]   =   min(std_x_sort);    %求最小的相邻 k 个值的标准方差和下标
i=index(j:(j+k-1));                          %求最小标准方差的相邻 k 个值在向量 x
                                             %中的方差
y=x(i);                                      %求最小标准方差的 k 个值
```

可以用下面语句来验证以上函数的正确性，如下：

```
%minvar_example.m
%验证已知 n 个数据中挑选出方差最小的 k 个数据的函数 minvar()
data = (1:10).^2;                 %数列  1 4 9 16 ...
```

```
index = randperm(10);            %1:10 的随机排列
x = data(index)                  %随机扰乱数据 data 的排列
[y i]=minvar(x,4)                %求数据 x 中方差最小的 4 个值
```

由上述语句得到输出代码如下：

```
x =
    16     4    36   100     9    81     1    25    64    49
y =
     1     4     9    16
i =
     7     2     5     1
```

6.3.2　协方差和相关系数矩阵

在数理统计中，协方差和相关函数是极其重要的随机变量的数字特征。若给定 n 维随机变量 $x=(x_1,x_2,\dots,x_n)$，定义

$$C_{ij}=E\left((x_i-E(x_i))(x_j-E(x_j))\right)$$

则称

$$C=\begin{pmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{pmatrix}$$

为随机变量 $x=(x_1,x_2,\dots,x_n)$的协方差矩阵。定义相关系数：

$$R_{ij}=\frac{C_{ij}}{\sqrt{C_{ii}*C_{jj}}}$$

则称

$$R=\begin{pmatrix} R_{11} & \cdots & R_{1n} \\ \vdots & \ddots & \vdots \\ R_{n1} & \cdots & R_{nn} \end{pmatrix}$$

为相关系数矩阵。

在 MATLAB 7.0 中，3 维随机变量的 100 次抽样数据应该表示为 100×3 的矩阵，其中矩阵每一列表示随机变量的一个分量，每一行代表一次抽样。

MATLAB 7.0 提供函数 cov()用于计算随机变量的协方差矩阵，其调用格式如下：

- C = cov(X)，计算 X 代表的随机变量的协方差矩阵。如果 X 是一个向量，返回向量的方差；如果 X 是矩阵，返回矩阵的协方差矩阵，其对角元是该列随机变量的方差；
- C = cov(x,y)，x 和 y 必须是具有相同长度的向量。相当于计算 C = cov([x y])；
- cov(X,1)，计算协方差时采用（5-2）式的方差，默认情况采用（5-1）式定义的无偏方

差；

- C = cov(x,y,1)，计算协方差时采用（5-2）式的方差，默认情况采用（5-1）式定义的无偏方差。

MATLAB 7.0 提供函数 corrcoef ()用于计算随机变量的相关系数矩阵，其调用格式如下。

- R = corrcoef(X)，返回 X 代表的随机变量的相关系数矩阵；
- R = corrcoef(x,y)，x 和 y 必须是具有相同长度的向量，相当于计算 C = corrcoef([x y])；
- [R,P]=corrcoef(...)，多返回一个 P 值用来检验不相关的几率。如果随机变量完全相关则 P 值为 0；如果 P 值很小，例如小于 0.05，则可以认为随机变量的相关性很大；
- [R,P,RLO,RUP]=corrcoef(...)，多返回与相关系数矩阵 R 有相同大小的矩阵 RLO 和 RUP，RLO 和 RUP 是相关系数在置信度为 95%情况的下界和上界。

例如，计算随机变量的系数相关矩阵，代码设置如下：

```
%corrcoef_example.m
%计算矩阵的相关系数
x = randn(10000,4);                          % 4 个完全无关随机变量
y(:,1) = x(:,1);
y(:,2) = x(:,2);
y(:,3) = 0.3*x(:,3) + 0.7*x(:,4);            %在第 3 个随机变量和第 4 个随机变量之间
y(:,4) = 0.2*x(:,3) + 0.8*x(:,4);            %引入相关性
[r,p] = corrcoef(y)                          %计算相关矩阵和不相关概率矩阵
[i,j] = find(p<0.05);                        %寻找相关的随机变量对
[i,j]                                        %显示相关的随机变量对的编号
```

由上述语句得到输出代码如下：

```
r =
    1.0000    0.0011    0.0009   -0.0017
    0.0011    1.0000    0.0019    0.0028
    0.0009    0.0019    1.0000    0.9870
   -0.0017    0.0028    0.9870    1.0000
p =
    1.0000    0.9163    0.9313    0.8642
    0.9163    1.0000    0.8496    0.7778
    0.9313    0.8496    1.0000         0
    0.8642    0.7778         0    1.0000
ans =
     4     3
     3     4
```

可见，只有随机变量的第 3 行和随机变量的第 4 行相关系数才比较大，这正是在程序中故意引入的相关性。

6.3.3　有限差分和梯度

在 MATLAB7.0 中使用函数 diff()来计算差分，其调用格式如下。

- Y = diff(X)，如果 X 是一个向量，则返回[X(2)-X(1) X(3)-X(2) ... X(n)-X(n-1)]；如果 X 是一个矩阵则返回[X(2:m,:)-X(1:m-1,:)]；
- Y = diff(X,n)，返回 n 阶差分，例如 diff(X,2)，与语句 diff(diff(X))效果相同；
- Y = diff(X,n,dim)，返回在 dim 维上的 n 阶差分。

例如，利用有限差分计算正弦函数的导数，代码设置如下：

```
%sin_diff.m
%正弦函数的导数
x=0:0.1:10;                              %自变量
y=sin(x);                                %正弦函数
y_der=diff(y)./diff(x);                  %导数等于有限差分之比(dy/dx)
hold on;
x_der=x(1:(end-1));                      %导数的自变量
plot(x,y,'b-');                          %画图
plot(x_der,y_der,'b-.');
axis([0 10 -1.2 1.4]);                   %设定坐标轴范围
legend('正弦函数','正弦函数的导数');     %添加图例
```

由上述语句得到如图 6-13 所示的示意图。

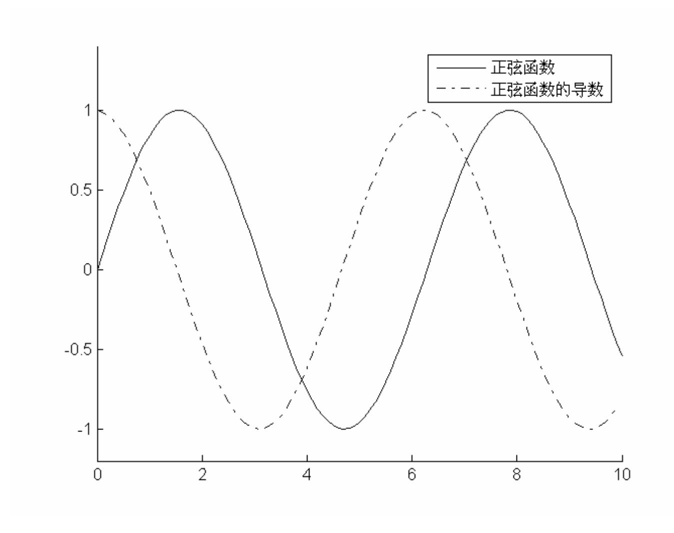


图 6-13　正弦函数及其导数

对于一个二元函数 F(x,y)，可以定义它的梯度如下：

$$\nabla F=\frac{\partial F}{\partial x}\hat{i}+\frac{\partial F}{\partial y}\hat{j}$$

该梯度代表 F 变化最快的方向。在 MATLAB 7.0 中，用函数 gradient()来计算梯度，其调用格式如下。

- FX = gradient(F)，F 是一个向量，返回 F 在 x 方向上的梯度，相当于计 F 的微分；
- [FX,FY] = gradient(F)，F 是二维矩阵，FX 是 F 在 x 方向的微分（列方向），FY 是 F 在 y 方向的微分（行方向），并假定自变量的间距是 1；
- [Fx,Fy,Fz,...] = gradient(F)，F 是 N 维矩阵，返回 N 个方向的微分值；
- [...] = gradient(F,h)，h 为一个标量，用于指定所有方向上自变量的间距；
- [...] = gradient(F,$h1$,$h2$,...)，用多个标量 $h1$,$h2$,…来指定各个方向上自变量的间距。

例如，计算二维高斯函数的梯度场，代码设置如下：

```
%gradient_example.m
%计算二维高斯函数的梯度场
v=-2:0.25:2;
[x,y]=meshgrid(v,v);                    %产生自变量 x,y
z= exp(-(x.^2+y.^2+0.5*x.*y));          %二维高斯函数
[px py]=gradient(z,0.25);               %梯度场
contour(v,v,z,4);                       %画 4 条等高线
hold on;
quiver(v,v,px,py);                      %画出梯度场
```

由上述语句得到如图 6-14 所示的示意图。图中椭圆是二维高斯函数的等高线，箭头的方向代表梯度的方向，箭头的大小代表梯度的模。由图可见梯度方向总是垂直于等高线的，这是梯度场的基本性质之一。

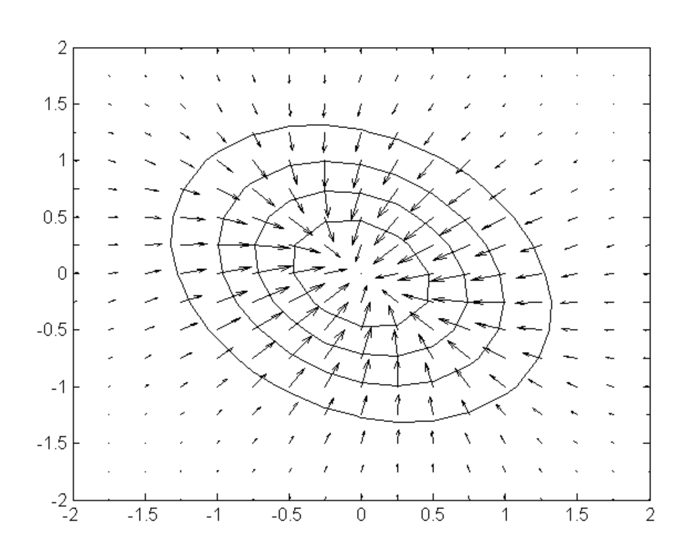


图 6-14　二维高斯函数的梯度场

6.3.4　信号滤波和卷积

MATLAB 7.0 中有关信号滤波和卷积的函数，如表 6-4 所示。

表 6-4　　　　信号滤波和卷积函数

函数名	功能描述
filter	一维数字滤波器
filter2	二维数字滤波器（将在本书的第 9 章详细介绍）
conv	一维卷积和多项式乘法
conv2	二维卷积（将在本书的第 9 章详细介绍）
convn	*N* 维卷积
deconv	反卷积和多项式除法
detrend	去除信号中的直流或者线形成分，主要用于 FFT 运算中

1．一维数字滤波

MATLAB 7.0 的一维数字滤波可以用函数 filter()来实现，它是直接 II 型线性差分方程组的解，可以用于实现 FIR 和 IIR 滤波。直接 II 型线性差分方程组的信号流程图如图 6-15 所示。

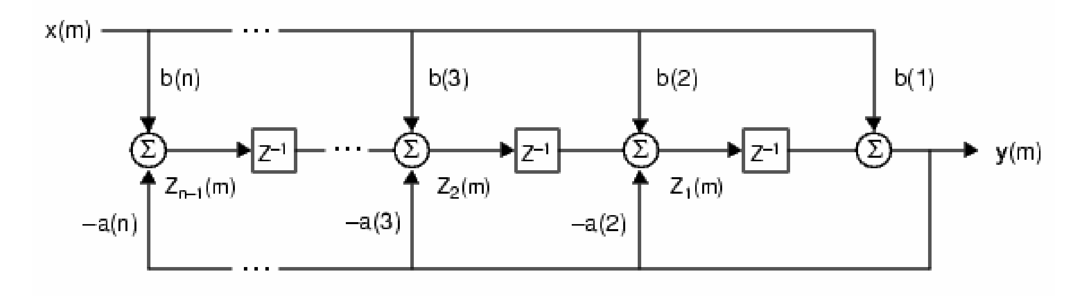


图 6-15　直接 II 型线形差分方程组

图 6-15 的线性差分方程组用如下表达式来表示：

$$y(n) = b(1)*x(n) + b(2)*x(n-1) + \ldots + b(n_b+1)*x(n-n_b)$$

$$- a(2)*y(n-1) - \ldots - a(n_a+1)*y(n-n_a)$$

其中 n_a 和 n_b 是向量 *b* 和 *a* 的长度。该方程是 *n*-1 阶的线性方程。该方程表现在 *Z* 域的形式为：

$$Y(z) = \frac{b(1)+b(2)Z^{-1}+\cdots+b(n_b+1)Z^{-n_b}}{1+a(2)z^{-1}+\cdots+a(n_a+1)z^{-n_a}}X(z)$$

函数 filter()的调用格式如下。

- *y* = filter(*b*,*a*,*X*)，*X* 为用于滤波的数据，向量 *a* 和 *b* 构造一个滤波器，*Y* 为数据 *X* 通过滤波器之后的值；
- [*y*,*zf*] = filter(*b*,*a*,*X*)，多返回一个表示数据延迟时间的量 *zf*。当 *X* 为向量时，*zf* 等于 max(length(*a*),length(*b*))-1；
- [*y*,*zf*] = filter(*b*,*a*,*X*,*zi*)，*zi* 为初始数据延迟，*zf* 等于最终数据延迟；
- *y* = filter(*b*,*a*,*X*,*zi*,*dim*)，在 *dim* 维上进行数据滤波。

如何得到滤波器的系数 *b* 和 *a* 涉及到滤波器设计问题，具体将在本书的第 9 章作出详细介绍。在有些情况下，可以很容易得到滤波器的系数 *b* 和 *a*。例如，一个 5 阶平均值滤波器：$y(n)=1/5*x(n) + 1/5*x(n-1) + 1/5*x(n-2) + 1/5*x(n-3) + 1/5*x(n-4)$，其系数 *a* 可以表示为 1，系

数 b 可以表示为[1/5 1/5 /15 1/5 1/5]。

下面通过一个示例来实现对带噪声的正弦信号进行平均值滤波，代码设置如下：

```
%filter_example.m
%对带噪声的正弦信号进行平均值滤波
t=0:0.1:10;                          %时间
n = 6*randn(size(t));                %高斯白噪声
x = 40*sin(t)+n;                     %在正弦信号中添加噪声
a = 1;                               %频率值滤波器的系数
b = [1/5 1/5 1/5 1/5 1/5];
y=filter(b,a,x);                     %滤波
plot(t,x,'b-');                      %画原始信号
hold on;
plot(t,y,'r:');                      %画滤波后的信号
axis([0 10 -60 65]);                 %设置坐标轴范围
xlabel('时间/ts(s)');
legend('原始数据','滤波后数据');     %添加图例
```

上述语句得到输出如图 6-16 所示。可见平均值滤波能有效去除信号的噪声。

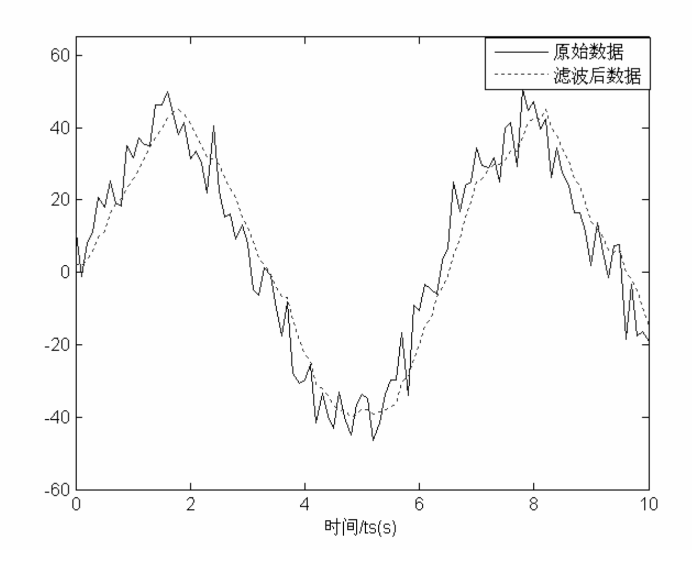


图 6-16　平均值滤波

2．信号卷积

向量 u 和 v 的卷积为 w，则记为 $w = u \otimes v$。如果 u，v 的长度分别为 m 和 n，则 w 的长度为 $m+n$-1。卷积的定义为：

$$w(k) = \sum_j u(j)v(k+1-j)$$

在上式中 j 可以使 $u(j)$和 $v(k+1-j)$有意义的任何值。卷积又一个重要的性质，即

fft(w)=fft(u)*fft(v)，在该式中函数 fft()表示信号的傅立叶变换。

MATLAB 7.0 使用函数 conv()来计算卷积。在前面第 6.1.3 小节，已经介绍了可以用该函数来计算多项式的乘法。函数 conv()计算卷积时利用快速傅立叶变换来实现，采用的步骤如下：

- U=fft(u)；
- V=fft(v)；
- w=ifft(u,v)。

例如，计算向量的卷积，代码设置如下：

```
%conv_example.m
%计算卷积的例子
u = ones(1,15);                %阶跃信号
v = zeros(1,25);
v(5:25) = 0:1/20:1;            %线性信号
w=conv(u,v);                   %卷积
subplot(3,1,1);                %画图
stem(u);
title('u');
subplot(3,1,2);
stem(v);
title('v');
subplot(3,1,3);
stem(w);
title('w');
```

由上述语句得到如图 6-17 所示的示意图。

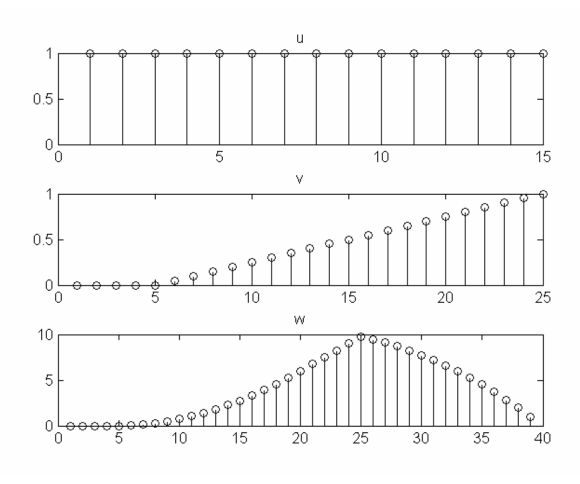


图 6-17　信号卷积

3．信号直流或线性成分去除

在做快速傅立叶变换之前经常需要去除信号中的直流成分或者线性成分。MATLAB 7.0提供的 detrend()函数就是用于实现去除信号中的直流成分或者线性成分的。其调用格式如下。

- *y* = detrend(*x*)，如果 *x* 是一个向量，从信号 *x* 中减去线性成分；如果 *x* 是一个矩阵，去除 *x* 所有列中的线性成分；
- *y* = detrend(*x*,'constant')，如果 *x* 是一个向量，从信号 *x* 中减去直流成分；如果 *x* 是一个矩阵，去除 *x* 所有列中的直流成分；
- *y* = detrend(*x*,'linear',*bp*)，从信号 *x* 中减去分段线性函数，分段线性函数的端点由输入 *bp* 决定。

例如，从信号中去除直流成分和线性成分，代码设置如下：

```
%detrend_example.m
%从信号中去除直流成分和线性成分
t = 0:0.04:5;                                   %时间
x = 2*t + 0.5*randn(size(t));                   %带线性成分的随机信号
x_no_linear=detrend(x);                         %去除去除线性成分
subplot(2,1,1);                                 %画图
hold on;
plot(t,x,'b-');
plot(t,x_no_linear,'b:');
axis([0 5 -2 14]);                              %设置坐标轴范围
title('从信号中去除线形成分');
legend('原始数据','去除线性成分的数据');          %添加图例
y = 3 + 0.5*randn(size(t));                     %带直流成分的随机信号
y_no_constant = detrend(y,'constant');          %去除直流成分
subplot(2,1,2);
hold on;
plot(t,y,'b-');
plot(t,y_no_constant,'b:');
axis([0 5 -2 8]);                               %设置坐标轴范围
title('从信号中去直流成分');
legend('原始数据','去除直流成分的数据');          %添加图例
```

由上述语句得到如图 6-18 所示的示意图。

6.3.5 傅立叶变换

对信号进行分析通常可以在时域中进行，也可以在频域中进行，时域分析方法和频域分析方法各有其优缺点。傅立叶变换是把信号从时域变换到频域，因此它在信号分析中占有极其重要的地位，如滤波器设计、频谱分析等方面。

傅立叶变换既可以对连续信号进行分析，也可以对离散信号进行分析。连续信号的傅立

叶变换实际上要计算傅立叶积分，这部分内容可以参见本书第 7 章。本小节只介绍离散傅立叶变换，即 Discrete Fourier Transform（DFT）。

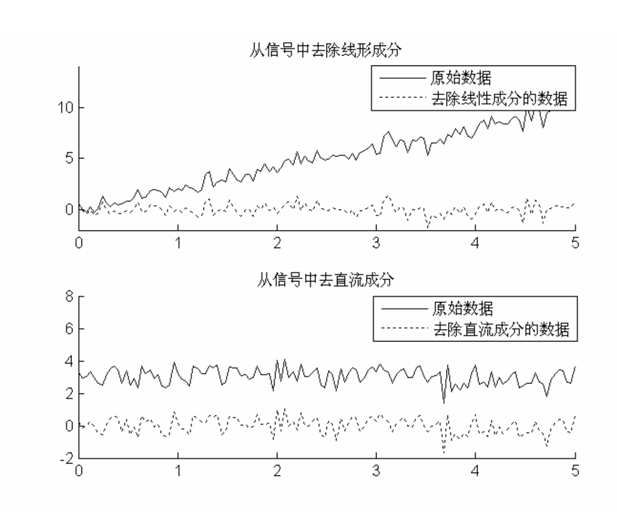


图 6-18　去除信号中的直流成分和线性成分

MATLAB 7.0 提供的有关傅立叶变换的函数如表 6-5 所示。

表 6-5　傅立叶变换函数

函数名	功能描述
fft	一维离散快速傅立叶变换
fft2	二维离散快速傅立叶变换
fftn	N 维离散快速傅立叶变换
ifft	一维离散快速傅立叶逆变换
ifft2	二维离散快速傅立叶逆变换
ifftn	N 维离散快速傅立叶逆变换
fftshift	把频谱的直流成分移到中间
nextpow2	返回大于或等于参数值的二次幂，满足 2^p>=abs(A)的最大整数

1．一维傅立叶变换和一维傅立叶逆变化

一维离散傅立叶变换的定义，如果有一个向量 $x(n)$，其离散傅立叶变换结果为 $X(k)$，$X(k)$ 与 $x(n)$的关系如下：

$$X(k)=\sum_{n=1}^{N}x(n)e^{-j2\pi(k-1)\frac{n-1}{N}},\quad 1\le k\le N,$$

$$x(n)=\frac{1}{N}\sum_{k=1}^{N}X(k)e^{j2\pi(k-1)\frac{n-1}{N}},\quad 1\le n\le N,$$

其中 N 为向量 $x(n)$的长度。

在 MATLAB 7.0 中一维离散傅立叶变换由函数 fft()来实现，其调用方式如下。

- Y = fft(X)，如果 X 是向量，返回向量 X 的傅立叶变换；如果 X 是矩阵，函数对矩阵 X 的每一列进行傅立叶变换；

- Y = fft(X,n)，用输入 n 指定傅立叶变换的长度。如果向量 X 的长度小于 n，则做傅立叶变换之前在向量的尾部添加 0 值，使得向量 X 为长度为 n；如果向量 X 的长度大于 n，则把向量 X 截断为长度为 n 的向量；
- Y = fft(X,[],*dim*)，在 *dim* 维上进行傅立叶变换；
- Y = fft(X,n,*dim*)，在 *dim* 维上进行傅立叶变换，并指定傅立叶变换的长度。

在 MATLAB 7.0 中一维离散傅立叶逆变换由函数 ifft()来实现，其调用方式基本与函数 fft()相同，只是添加一个选项，其调用方式如下。

- y = ifft(..., 'symmetric')，把频谱 $X(k)$看成是共轭对称。该选项在 $X(k)$不是严格共轭对称时有效；
- y = ifft(..., 'nonsymmetric')，该选项为默认选项。

与傅立叶变换函数经常使用的函数是 fftshift()函数。如图 6-19 所示，信号 $x(n)$经过 FFT 后得到频谱 $X(k)$，$X(k)$的头部和尾部代表信号的低频分量大小，$X(k)$的中间则代表高频部分。习惯上画频率响应曲线时把直流成分放在曲线的中间，而把高频部分放在曲线的头部和尾部。因此频谱 $X(k)$不可以直接用于画频率响应曲线。为了画图时方便，MATLAB 7.0 提供函数 fftshift()用于把 FFT 变换的结果频谱 $X(k)$转换为适合画图显示的格式。对一维信号操作时，fftshift()函数把信号分为左半部分和右半部分，然后交换其左右部分。

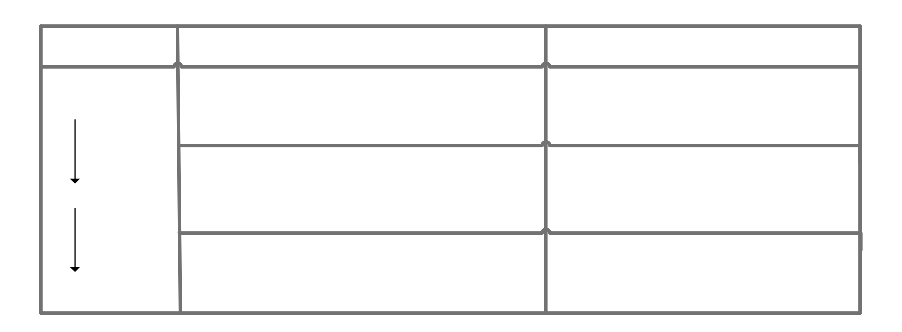


图 6-19 N 为偶数时 fftshift()函数的功能示意图

fftshift()函数不但可以用于一维 FFT 结果的调整，还可以用于二维 FFT 甚至多维 FFT 的调整，其调用格式如下。

- Y = fftshift(X)，如果 X 是向量，交换向量的左部分和右部分；如果 X 是矩阵，矩阵的第一象限和第三象限交换，第二象限和第四象限交换；
- Y = fftshift(X,*dim*)，在某一维上交换数据的左部分和右部分。

例如，计算单位冲激信号经过一个带阻滤波器前后的频谱，代码设置如下：

```
%fliter_example.m
%单位冲击信号通过带阻滤波器
t = 1:40;                                  %信号的时间
x =zeros(size(t));
x(1) = 1;                                  %产生单位冲击信号
```

```
[b,a] = butter(10,[0.3 0.7],'stop');                    %设计带阻滤波器
y = filter(b,a,x);                                      %滤波
hold on;                                                %画图滤波前后的时域数据
stem(t,x,'marker','o');
stem(t,y,'marker','.');
xlabel('时间/ts(s)');
legend('原始数据','滤波后数据');
fx=fft(x);                                              %对单位冲击信号进行傅立叶变换
fx=fftshift(fx);
fy=fft(y);                                              %滤波后的信号进行傅立叶变换
fy=fftshift(fy);
figure;
subplot(2,1,1);                                         %画图显示滤波前后的数据频谱
f=(t-20)/20;
plot(f,abs(fx),'b-',f,abs(fy),'b-.');                   %画信号的幅频曲线
xlabel('数字频率/\pi(rad)');
title('幅频曲线');
legend('原始数据的幅频曲线谱','滤波后数据的幅频曲线');
subplot(2,1,2);
plot(f,angle(fx),'b-',f,angle(fy),'b-.');               %画信号的相频曲线
xlabel('数字频率/\pi(rad)');
title('相频曲线');
legend('原始数据的相频曲线','滤波后数据的相频曲线');
```

由上述程序得到输出如图 6-20 所示。其中用到滤波器设计的函数 butter()，关于该函数的详细内容将在第 9 章中介绍。

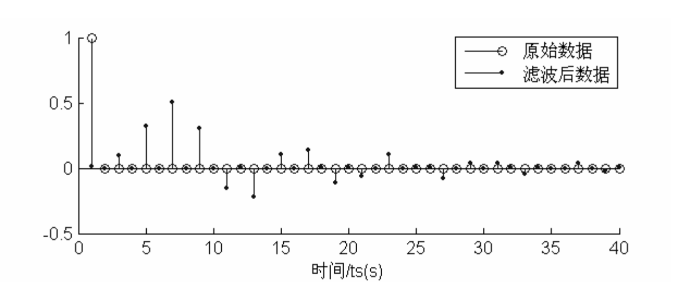


图 6-20　单位冲击信号通过带阻滤波器的时域信号

由图 6-21 可见，单位冲激信号的频谱是一个常数，它通过一个带阻滤波器后的频谱等于带通滤波器的频率响应曲线。这个结果验证了一个信号通过一个线性系统后的频谱等于信号频谱乘以线性系统的频率响应的结论。

2．二维傅立叶变换

在图像处理中经常使用二维傅立叶变换来进行图像滤波等操作。在 MATLAB7.0 中二维傅立叶变换用函数 fft2()来实现。函数 fft2()可以看成是对信号进行两次一维傅立叶变换，即 fft2(*X*)=fft(fft(*X*).').'。函数 fft2()的调用格式如下：

- *Y* = fft2(*X*)，*X* 是矩阵，对矩阵 *X* 进行二维傅立叶变换；
- *Y* = fft2(*X*,*m*,*n*)，*m* 和 *n* 指定傅立叶变换的长度。如果小于该长度则在信号的尾部添加 0 值；如果大于该长度则截断信号。

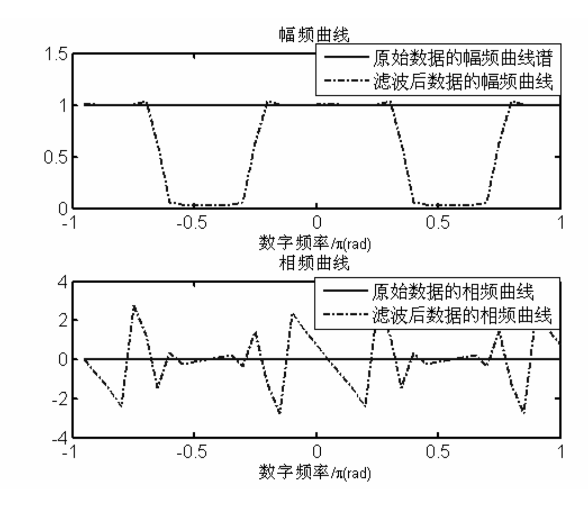


图 6-21　单位冲击信号通过带阻滤波器的频域信号

例如，分析图 6-22 的频谱，代码设置如下：

```
%fft2_example.m
%分析图像的频谱
img = imread('coins.png');                              %读图像文件
f_img = fft2(double(img));                              %二维 fft 变换
f_img = fftshift(f_img);                                %把直流成分移动频谱的中间
imshow(img);                                            %显示图像
figure;
f_img_abs = abs(f_img);                                 %得到频谱幅度
f_img_abs = (f_img_abs-min(min(f_img_abs)))./ ..       . %把频谱幅度变换到[0,255]范围内
    (max(max(f_img_abs))-min(min(f_img_abs)))*255;
imshow(f_img_abs);                                      %显示频谱幅度
title('图像的幅频分布');
```



图 6-22　硬币

由上述语句得到图 6-22 的幅频图，如图 6-23 所示。

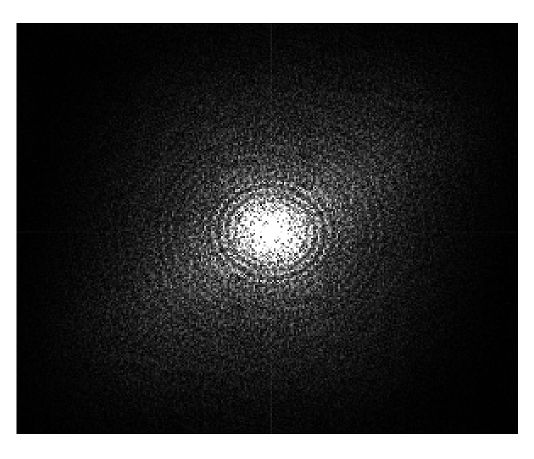


图 6-23　幅频图

6.4　功能函数

功能函数就是可以用其他函数作为输入变量的函数，例如一个可以画函数图像的函数 fplot()。其调用格式为 fplot(@*fun*,[-pi,pi])。输入变量@*fun* 为需要画成图的函数的函数句柄。MATLAB 7.0 的功能函数都在目录“\toolbox\matlab\funfun”下。

6.4.1　函数的表示

在 MATLAB 7.0 中函数可以用 M 文件、匿名函数和 inline 对象来表示。其中匿名函数是 MATLAB 7.0 的新特性。例如，要表示函数 $y=\mathrm{f}(x)=\dfrac{2}{1+e^{-x}}+\dfrac{3}{1+e^{-2x}}$

可以通过如下 M 文件来表示上面的函数：

```
%customized_fun.m
%用户自己定义的函数
function y=customized_fun(x)
y=2./(1+exp(-x))+3./(1+exp(-2*x));
```

把 customized_fun.m 文件放在 MATLAB 7.0 的路径下，例如“\work”目录下，就可以使用该函数了。例如在命令行中输入如下代码：

```
customized_fun(magic(4))
```

由上述语句得到输出代码如下：

```
ans =
     5.0000     4.7076     4.8977     5.0000
     4.9865     5.0000     4.9999     4.9993
     4.9998     4.9982     4.9950     5.0000
     4.9630     5.0000     5.0000     4.1045
```

MATLAB 7.0 中也允许使用匿名函数来表示一个函数，例如可以用下面的语句表示上述函数：

```
%anonymous_fun.m
%创建匿名函数和使用匿名函数的例子
fh = @(x)2./(1+exp(-x))+3./(1+exp(-2*x));
fh(magic(4))
```

由上述语句得到输出代码如下：

```
ans =
     5.0000     4.7076     4.8977     5.0000
     4.9865     5.0000     4.9999     4.9993
     4.9998     4.9982     4.9950     5.0000
     4.9630     5.0000     5.0000     4.1045
```

在 MATLAB 7.0 以前的版本中经常使用 inline()函数来把一个字符串转换为一个 inline 对象，该 inline 对象可以用来表示函数。MATLAB 7.0 仍然可以使用 inline 对象，例如可以用如下的代码表示上述函数：

```
%inline_fun.m
%创建 inline 函数和使用 inline 函数的例子
g = inline('2./(1+exp(-x))+3./(1+exp(-2*x))');
g(magic(4))
```

由上述语句得到的输出结果与使用匿名函数时的输出结果是一样的。有些功能函数还接受使用字符串来表示函数，实际上相当于功能函数内部使用 inline()函数把一个字符串转换为 inline 对象。

6.4.2 函数画图

MATLAB 7.0 提供的用于函数画图的函数如表 6-6 所示。

表 6-6　　函数画图的函数

函数名	功能描述
fplot	函数画图
ezplot	易用的函数画图
ezplot3	易用的三维函数画图
ezpolar	易用的极坐标画图
ezcontour	易用的等高线画图
ezmesh	易用的三维网格画图
ezmeshc	易用的混合网格和等高线画图
ezsurf	易用的三维彩色表面画图
ezsurfc	易用的混合表面和等高线画图

函数画图函数的用法比较简单，而且用法相似。下面就以常用的 fplot()函数为例来介绍这些函数的用法。

fplot()的基本调用格式如下。

- fplot(*function*,*limits*)，*function* 为待画图的函数，*limits* 是横坐标数值范围[*xmin xmax*]，或者是横纵坐标数值范围[*xmin xmax ymin ymax*]；
- fplot(*function*,*limits*,*LineSpec*)，*LineSpec* 指定画图的线条属性，与 plot()函数的线条属性用法一致；
- fplot(*function*,*limits*,*tol*)，*tol* 指定画图相对精度，默认值时 2e-3；
- fplot(*function*,*limits*,*tol*,*LineSpec*)，指定画图的线条属性和画图相对精度；
- fplot(*function*,*limits*,*n*)，用于指定最少画图点数。函数画图时至少画 n+1 个点。默认值是 1。

例如，画函数 $y = \cos(\frac{x+1}{x^2+1})$ 在区间[0,10]上的函数图像，代码设置如下：

```
%fplot_example.m
%函数画图实例
f = @(x)cos((x+1)./(x.^2+1));                    %匿名函数
fplot(f,[-5 5],1e-4,'r-');                       %函数画图
title('函数 y=cos((x+1)/(x^2+1))');
xlabel('x');
ylabel('y');
grid;
```

由上述语句得到结果如图 6-24 所示的示意图。

6.4.3　函数最小值和零点

求函数最小值和函数的零点是工程上常见的问题，MATLAB 7.0 提供一些函数用于解决这类问题。MATLAB 7.0 提供的函数如表 6-7 所示。

表 6-7　　求函数最小值和零点

函数名	功能描述
fminbnd	求一元函数的给定区间内的最小值
fminsearch	求多元函数在给定点附近的局部最小值
fzero	求一元函数的零点
optimset	设定求最小值时的优化器参数
optimget	得到求最小值时的优化器参数

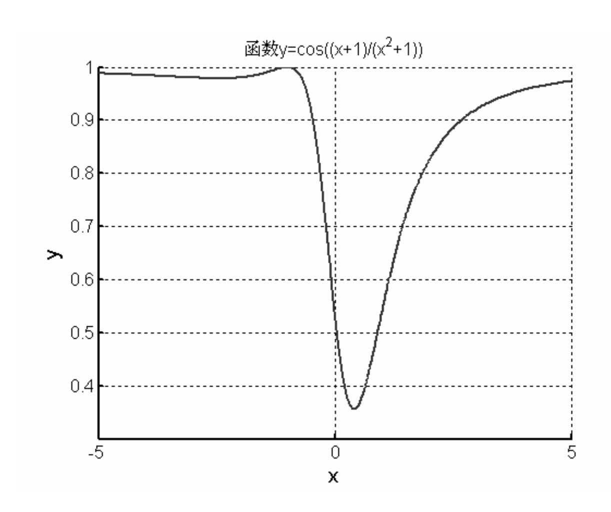


图 6-24　函数画图结果

> 小技巧：如果需要求函数的最大值，只要先把原函数乘上一个-1 得到一个新函数，然后求这个新函数的最小值。则新函数最小值的自变量取值与原函数最大值的的自变量取值相同。

1．一元函数最小值

一元函数在给定区间内的最小值问题可以用函数 fminbnd()来实现，其调用格式如下。

- *x* = fminbnd(*fun*, ,*x*1,*x*2)，在区间[*x*1 *x*2]内寻找函数最小值。*fun* 为 M 文件的函数句柄或者匿名函数，*x* 为最小值的自变量取值；
- *x* = fminbnd(*fun*,*x*1,*x*2,*options*)，使用 *options* 选项来指定的优化器的参数。*options* 可以使用函数 optimset()来设定；
- [*x*,*fval*] = fminbnd(...)，多返回一个输出变量最小值 *fval*。

> 注意：函数 fminbnd()只能用于连续函数，并且只给出局部最小值。当最小值是在指定区间边界上时，函数收敛的速度很慢。

例如，求正弦函数在[0,10]内的最小值，代码设置如下：

```
%fminbnd_example1.m
%求正弦函数在[0 10]内的最小值
x=fminbnd(@sin,0,10)
```

由上述语句得到输出代码如下：

```
x =
```

```
    4.7124
```

如果要显示计算函数最小值的过程，则可以使用 *options* 参数来设定，代码设置如下：

```
%fminbnd_example2.m
%求正弦函数在[0 10]内的最小值
x=fminbnd(@sin,0,10,optimset('Display','iter'))
```

由上述语句得到输出代码如下：

```
Func-count     x          f(x)         Procedure
    1       3.81966    -0.627289       initial
    2       6.18034    -0.102664       golden
    3       2.36068     0.703928       golden
    4       4.62594    -0.996266       parabolic
    5       4.74595    -0.999437       parabolic
    6       4.71433    -0.999998       parabolic
    7       4.71238           -1       parabolic
    8       4.71242           -1       parabolic
    9       4.71235           -1       parabolic
 Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-004
x =
    4.7124
```

2．求多元函数的最小值

求多元函数的最小值用函数 fminsearch()来实现。使用该函数时必须指定开始矢量 *X*0，此函数返回一个在矢量 *X*0 附近的局部最小值。其调用格式如下。

- *x* = fminsearch(*fun*,*x*0)，在矢量 *X*0 附近寻找的局部最小值。*fun* 为 M 文件的函数句柄或者匿名函数，*x* 为最小值的自变量取值；
- *x* = fminsearch(*fun*,*x*0,*options*)，使用 *options* 选项来指定的优化器的参数。*options* 可以使用函数 optimset()来设定；
- [*x*,*fval*] = fminsearch(...)，多返回一个输出变量最小值 *fval*。

例如，求二维函数 $f(x)=100(x_2-x_1^2)^2+(1-x_1)^2$ 的局部最小值，代码设置如下：

```
%fminsearch_example.m
%寻找二维最小值
banana = @(x)100*(x(2)-x(1)^2)^2+(1-x(1))^2;
[x,fval] = fminsearch(banana,[-1.2, 1])
```

由上述语句得到输出代码如下：

```
x =
    1.0000    1.0000
fval =
  8.1777e-010
```

3．求一元函数的零点

可以使用函数 fzero()来求一元函数的零点。寻找一元函数零点时，可以指定一个开始点，或者指定一个区间。当指定一个开始点时，此函数在开始点附近寻找一个使函数值变号的区间，如果没有找到这样的区间，则函数返回 NaN。如果知道函数零点所在的区间，则可以使用一个包含两个元素的向量来指定此区间。fzero()函数的调用格式如下。

- *x* = fzero(*fun*,*x*0)，在 *x*0 点附近寻找函数的零点。返回值 *x* 是零点的自变量值；
- *x* = fzero(*fun*,*x*0,*options*)，用 *options* 参数指定寻找零点的优化器参数；
- [*x*,*fval*] = fzero(...)，多返回一个输出变量 *fval*，表示函数 *fun* 在自变量为 *x* 时的函数值；
- [*x*,*fval*,*exitflag*] = fzero(...)，多返回一个函数运行返回状态标志。

例如，用函数 fzero()来解一元非线性方程 $\frac{1}{(x+4)^2+1}+\frac{1}{(x-4)^2+1}=\frac{1}{2}$。该问题等价于求函数 $f(x)=\frac{1}{(x+4)^2+1}+\frac{1}{(x-4)^2+1}-\frac{1}{2}$ 的零点，函数 f(*x*)的曲线图形如图 6-25 所示。

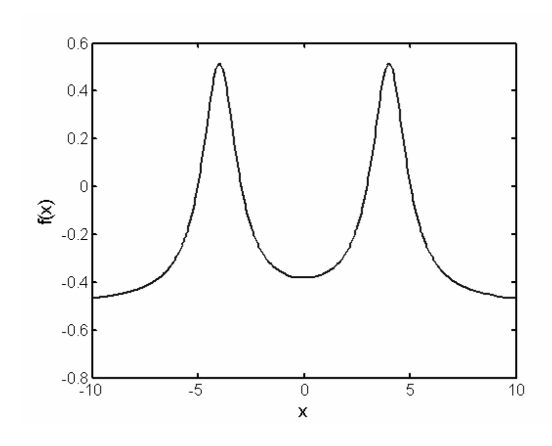


图 6-25　函数 f(*x*)的曲线图

求解方程的代码设置如下：

```
%fzero_example.m
%求解一元非线性方程
f = @(x) 1./((x+4).^2+1)+1./((x-4).^2+1)-0.5;                    %用匿名函数来表示函数
fplot(f,[-10 10]);                                                %画出函数的图
xlabel('x');
ylabel('f(x)');
x1 = fzero(f,-5)                                                  %求某个点附近的零点
x2 = fzero(f,-3)
x3 = fzero(f,3)
x4 = fzero(f,5)
x1_region = fzero(f,[-10,-3])                                     %求某个区间内的零点
```

```
x2_region = fzero(f,[-3,0])
x3_region = fzero(f,[0 3])
x4_region = fzero(f,[3 10])
```

由上述语句得到输出代码如下：

```
x1 =
   -5.0246
x2 =
   -2.9587
x3 =
    2.9587
x4 =
    5.0246
x1_region =
   -5.0246
x2_region =
   -2.9587
x3_region =
    2.9587
x4_region =
    5.0246
```

> 注意：fzero()函数只能返回一个局部零点，不能寻找出所有的零点。此外，fzero()函数的收敛速度与开始点的选取或者区间的选取有关系，尽可能先猜出零点的大概范围，然后在该范围内寻找零点。

4．优化器参数

在求一元函数、多元函数最小值以及求一元函数零点时，都可以设定求优化器的参数。设定优化器参数的函数为 optimset()，其调用格式如下：

- *options* = optimset(*'param1',value1,'param2',value2,...*)，用参数名和对用的参数值设定优化器的参数。返回值 *options* 是一个结构体；
- optimset，显示优化器的所有参数名和有效的参数值；
- *options* = optimset，返回一个优化器的结构体，该结构体的所有属性局为空矩阵；
- *options* = optimset(*optimfun*)，返回求最小值函数 *optimfun* 对应的优化器参数；
- *options* = optimset(*oldopts,'param1',value1,...*)，在原优化器参数 *oldopts* 的基础上，改动某些优化器参数；
- *options* = optimset(*oldopts,newopts*)，用 *oldopts* 和 *newopts* 组和得到最终的优化器参数。*newopts* 的所有非空参数覆盖 *oldopts* 中的值。

函数 optimset()中常用的优化器参数如表 6-8 所示。

表 6-8　　　　优化器参数

参数名	有效参数值	功能描述
Display	’off’，’iter’，’final’和’notify’	’off’：不显示计算结果 ’iter’：显示每个迭代步骤的计算结果 ’final’：只显示最终结果，该选项为默认值 ’notify’：只在计算不收敛时显示计算结果
FunValCheck	’off’和’on’	’off’：不对输入函数的返回值进行检查，该选项为默认值 ’on’：如果输入函数的返回值为复数或者 NaN 则显示警告信息
MaxFunEvals	正整数	最大允许的函数赋值次数
MaxIter	正整数	最大允许的迭代次数
OutputFcn	用户定义的函数句柄或者空矩阵	空矩阵：迭代过程采用 MATLAB 7.0 自带的函数 用户自定义函数句柄：用该函数替换 MATLAB 7.0 自带的函数
TolFun	正标量	函数值的截断阈值
TolX	正标量	自变量的截断阈值

如果要得到最小优化器的参数，则可以使用函数 optimget()，其调用格式如下：

- *val* = optimget(*options,’param’*)，返回最小优化器参数*’param* 的值；
- *val* = optimget(*options,’param’,default*)，返回最小优化器参数*’param* 的值，如果该值为空则返回 *default*。

6.4.4　数值积分

MATLAB 7.0 提供的数值积分函数如表 6-9 所示。

表 6-9　　　　数值积分函数

参数名	功能描述
quad	一元函数的数值积分，采用自适应 Simpson 方法
quadl	一元函数的数值积分，采用自适应 Lobatto 方法
quadv	一元函数的矢量数值积分
dblquad	二重积分
triplequad	三重积分

1．一元函数的数值积分

MATLAB 7.0 提供了两个函数 quad()和 quadl()计算一元函数的积分。函数 quad()采用低阶的自适应递归 Simpson 方法，函数 quadl()则采用高阶的自适应 Lobatto 方法。函数 quad()函数的调用格式如下：

- *q* = quad(*fun,a,b*)，计算函数 *fun* 在[*a b*]区间内的定积分。*fun* 为函数句柄，*a* 和 *b* 都是标量，它们分别是积分区间的下界和上界；
- *q* = quad(*fun,a,b,tol*)，以绝对误差容限 *tol* 计算函数 *fun* 在[*a b*]区间内的定积分，取代 MATLAB 7.0 中默认的绝对误差容限值 10^{-6}；
- *q* = quad(*fun,a,b,tol,trace*)，当 *trace* 为非零值时，显示计算定积分的迭代过程中向量[*fcnt a b-a Q*]的值；
- [*q,fcnt*] = quadl(*fun,a,b,...*)，多返回一个输出变量 *fcnt*，表示计算定积分过程中计算函数

值的次数。

例如，求归一化高斯函数的在区间[-1 1]上的定积分，代码设置如下：

```
%quad_exam.m
%求归一化高斯函数的在区间[-1 1]上的定积分,并得到积分过程的中间结点
y=@(x)1/sqrt(pi)*exp(-x.^2);                    %归一化高斯函数
quad(y,-1,1,2e-6,1)                             %求定积分，并显示中间迭代过程
fplot(y,[-1 1],'b');                            %画出函数
hold on;
%跟踪数据（运行完上面程序后，可以在命令行中复制这些数据）
trace = [ 9    -1.0000000000    5.43160000e-001    0.1804679399;
        11    -1.0000000000    2.71580000e-001    0.0728222057;
        13    -0.7284200000    2.71580000e-001    0.1076454255;
        15    -0.4568400000    9.13680000e-001    0.4817487615;
        17    -0.4568400000    4.56840000e-001    0.2408826755;
        19    -0.4568400000    2.28420000e-001    0.1142172651;
        21    -0.2284200000    2.28420000e-001    0.1266655031;
        23     0.0000000000    4.56840000e-001    0.2408826755;
        25     0.0000000000    2.28420000e-001    0.1266655031;
        27     0.2284200000    2.28420000e-001    0.1142172651;
        29     0.4568400000    5.43160000e-001    0.1804679399;
        31     0.4568400000    2.71580000e-001    0.1076454255;
        33     0.7284200000    2.71580000e-001    0.0728222057];
x1 = trace(:,2);                                %积分过程的中间结点
y1 = y(x1);                                     %中间结点的函数值
plot(x1,y1,'ro');                               %画图
xlabel('x');
ylabel('y');
legend('高斯函数','求积分过程的中间结点');
```

由上述语句得到输出代码如下：

```
ans =
    0.84270079883874
```

积分过程如图 6-26 所示。

函数 quadl()的调用格式与函数 quad()相同，可以参见上面 quad()函数的用法。

在一维积分的过程中，有可能得到 3 种警告信息：

- 'Minimum step size reached'，已经达到了最小步长。这意味着积分的迭代区间比给定的定积分区间的截断误差值小。一般来说，这表明被积函数的可积性是奇异的；
- 'Maximum function count exceeded'，计算函数值的次数超过 10000，一般来说，这表明被积函数的可积性是奇异的；
- 'Infinite or Not-a-Number function value encountered'，积分过程中出现浮点数溢出或者被 0 除。

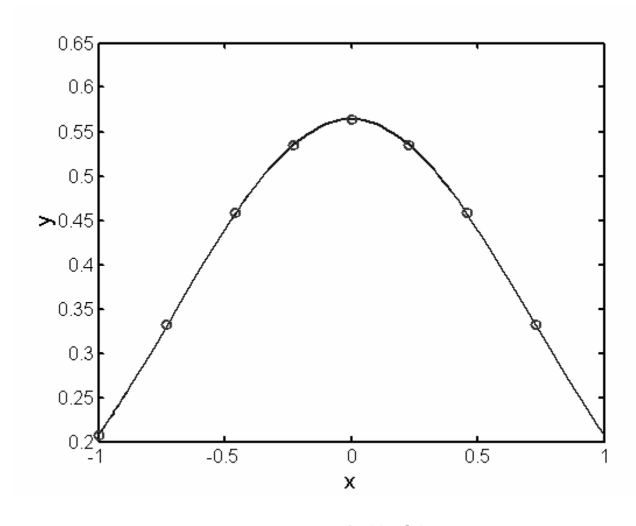


图 6-26　积分过程

2．矢量积分

矢量积分相当于多个一元定积分，例如求 $\int_{-1}^{1}\frac{1}{\sqrt{2\pi}*n}\exp(-\frac{x^2}{2n^2})\mathrm{d}x,\quad n=1,2,3,4,5$ 就可以用矢量积分，代码设置如下：

```
%quadv_exam.m
%求矢量积分
y=@(x,n)1./(sqrt(2*pi).*(1:n)).*exp(-x.^2./(2*(1:n).^2));         %归一化高斯函数
quadv(@(x)y(x,5),-1,1)
```

由上述语句得到输出代码如下：

```
ans =
     0.6827        0.3829        0.2611        0.1974        0.1585
```

矢量积分的结果是一个向量，其每一元素的值为一个一元函数定积分的值。

3．二重和三重积分

二重积分的形式如下：

$$Q=\int_{y_{\min}}^{y_{\max}}\int_{x_{\min}}^{x_{\max}} f(x,y)\mathrm{d}x\mathrm{d}y$$

MATLAB 7.0 中使用函数 dblquad()来计算二重积分。该函数先计算内积分值，然后利用内积分的中间结果来计算二重积分。根据 d*x*d*y* 的顺序，称 *x* 为内积分变量，*y* 为外积分变量。函数 dblquad()的基本格式如下：

- *q* = dblquad(*fun*,*xmin*,*xmax*,*ymin*,*ymax*)，计算二元函数 *fun* 在矩形区域[*xmin*,*xmax*, *ymin*,*ymax*]上的二重积分。*fun* 为函数句柄，*xmin*、*ymin*、*xmax* 和 *ymax* 都是标量，它们分别是积分区间的下界和上界；

- q = dblquad(*fun*,*xmin*,*xmax*,*ymin*,*ymax*,*tol*)，用 *tol* 指定绝对计算精度；
- q = dblquad(*fun*,*xmin*,*xmax*,*ymin*,*ymax*,*tol*,*method*)，用 *method* 指定计算一维积分时采用的函数。MATLAB7.0 默认采用函数 quad()来计算一维积分，可以设 *method*=@quadl，从而采用函数 quadl()来计算一维积分。用户还可以编写自己的一维积分函数以用于计算二维积分，该函数的调用格式必须与函数 quad()一致。

例如，计算二维高斯函数在矩形区间[-1 1 -1 1]上的二重积分，代码设置如下:

```
%dblquad_exam.m
%计算二维高斯函数在矩形区间[-1 1 -1 1]上的二重积分
f=@(x,y)1/sqrt(pi)*exp(-x.^2)*1/sqrt(pi)*exp(-y.^2);        %归一化高斯函数
dblquad(f,-1,1,-1,1,1e-6,@quadl)
```

由上述语句得到输出代码如下：

```
ans =
    0.71014343481071
```

函数 dblquad()处理的都是矩形积分区域。若要计算非矩形的积分区间的二重积分，可以用一个大的矩形积分区域包含非矩形的积分区间，然后在非矩形的积分区间之外的区域上把二元函数的值取零。

例如，计算二维高斯函数在圆形区域 $\sqrt{x^2+y^2}<1$ 上的二重积分，代码设置如下：

```
%dblquad_exam2.m
%计算二维高斯函数在圆形区域 sqrt(x.^2+y.^2)<1 上的二重积分
%在圆形区域外填充 0 的归一化高斯函数
f=@(x,y)(1/sqrt(pi)*exp(-x.^2)*1/sqrt(pi)*exp(-y.^2)).*(sqrt(x.^2+y.^2)<=1);
dblquad(f,-2,2,-2,2,1e-6,@quadl)
```

由上述语句得到输出代码如下：

```
ans =
      0.6321
```

三重积分的的形式如下：

$$Q=\int_{z_{\min}}^{z_{\max}}\int_{y_{\min}}^{y_{\max}}\int_{x_{\min}}^{x_{\max}} f(x,y,z)dxdydz$$

三重积分可以用函数 triplequad()来实现，其用法与二重积分类似。

6.4.5　在功能函数中使用含参函数

在很多情况下，例如要求不同 a 和 b 情况下，计算函数 $\mathrm{f}(x)=e^x+a*x-b$ 的零点。这时在功能函数中就要使用含参函数，它有两种方法：使用嵌套函数和使用匿名函数。

1．用嵌套函数提供函数参数

一个使用含参函数的方法是编写一个 M 文件，该文件首先把含参函数的参数当作输入，

然后调用功能函数来处理含参函数，最后把含参函数以嵌套函数的方式包含在 M 文件中。例如，求函数 $f(x) = e^x + a*x - b$ 的零点，代码设置如下：

```
%fzero_nestedfun.m
%用嵌套函数的方法实现处理含参函数
function y = fzero_nestedfun(a,b,x0)
%a,b 是含参函数的参数值
%x0 是求函数零点的开始点
options = optimset('Display', 'off');            %关闭显示
y = fzero(@para_fun,x0,options)                  %求函数零点
    function y=para_fun(x)                       %含参函数
        y=exp(x)+a*x-b;
   end
end
```

例如求 a=1,b=[-10 10]时，含参函数的零点取值随着参数 b 变化的关系，可以通过如下程序来实现：

```
%fzero_nestedfun_exam.m
%使用含参函数的例子
clear;
a=1;                                             %参数取值
b=-10:10;
x = zeros(size(b));
for i=1:length(b),
    x(i) = fzero_nestedfun(a,b(i),0);            %求函数零点
end;
plot(b,x);                                       %画图
xlabel('b');
ylabel('零点');
```

由上述语句得到如图 6-27 所示的示意图。

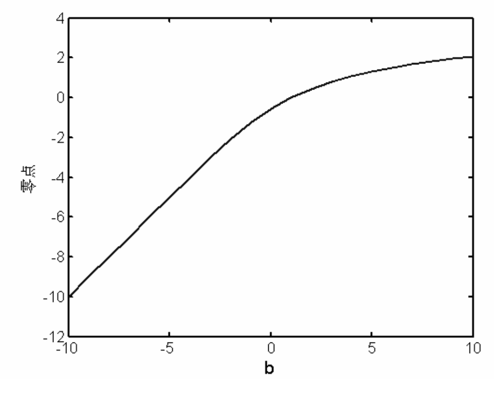


图 6-27　零点轨迹

2．用匿名函数提供函数参数

使用含参函数还可以利通过匿名函数来实现。由于与函数 fzero()一样，一般功能函数只接受一个自变量，因此函数的参数在使用之前必须先赋值。具体步骤如下：

- 首先创建一个含参函数，保存为 M 文件格式。函数输入为自变量 x 以及函数参数；
- 在调用功能函数的 M 文件中给参数赋值；
- 用含参函数创建匿名函数；
- 把匿名函数句柄传递给功能函数计算。

例如，用匿名函数来实现上小节的例子，首先建立含参函数，其代码设置如下：

```
%指数函数与线性函数的交点
function y=exp_linear(x,a,b)
    y=exp(x)+a*x-b;
```

然后，用如下代码来实现计算函数零点：

```
%fzero_anonymousfun.m
%用匿名函数提供函数参数
clear;
a=1;                                    %参数取值
b=-10:10;
x = zeros(size(b));
for i=1:length(b),
    f = @(x) exp_linear(x,a,b(i));
    x(i) = fzero(f,0);                  %求函数零点
end;
plot(b,x);                              %画图
xlabel('b');
ylabel('零点');
```

由上述语句得到的输出图形与图 6-27 是一致的。

6.5　微分方程组数值解

在 MATLAB 7.0 中可以计算 3 类微分方程数值解问题，即常微分方程组的初值问题、延迟微分方组和常微分方程组的边界问题。

6.5.1　常微分方程组的初值问题

MATLAB 7.0 可以解决 3 种常微分方程组的初值问题，即显式常微分方程组、线性隐式常微分方程组和完全隐式常微分组。显式常微分方程组的初值问题的形式如下：

$$y' = f(t, y) \qquad y(t_0) = y_0$$

其中 y’，y 和 y_0 为向量，故可以代表常微分方程组。显式常微分方程组的初值问题即已

知常微分方程的初值 $y(0)$，求 y 随时间的变化 $y(t)$。

线性隐式常微分方程组可以表述为如下形式：

$$M(t,y)y' = f(t,y) \qquad y(t_0) = y_0$$

该形式也称为加权常微分方程组。

完全隐式常微分组可以表述为为如下形式：

$$f(t,y,y') = 0 \qquad y(t_0) = y_0$$

以上介绍的都是一阶常微分方程组问题，如果遇到高阶常微分方程则可以先把高阶常微分方程转换为一阶常微分方程组。一个任意的高阶常微分方程可以表述如下：

$$y^{(n)} = f(t,y,y',\ldots,y^{(n-1)})$$

上式可以化为如下一阶常微分方程组：

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= y_3 \\ &\vdots \\ y_n' &= f(t,y_1,y_2,\ldots,y_n) \end{aligned}$$

1．显式常微分方程组的解法

由于显式常微分方程组的多样性，需要根据显式常微分方程组的特点采用相应的解法来处理。MATLAB7.0 给出 7 种解法，分别由 7 个不同的函数来实现，它们是函数 ode45()、ode23()，ode113()、ode15s()、ode23s()、ode23t()和 ode23tb()。

在介绍这些函数之前，首先介绍一个概念——刚性(stiffness)。定性地讲，对一个常微分方程组，如果其 Jocabian 矩阵的特征值相差十分悬殊，那么这个方程组就称为刚性方程组。对于刚性方程组，为了保持解法的稳定，步长选取很困难。有些解法不能用于解刚性方程组，而有些解法对方程组的刚性要求不严，可以用于解刚性问题。表 6-10 对 MATLAB 7.0 中的 7 个解法进行了对比。

表 6-10　　常微分方程组解法对比

函数名	采用算法	精度	适用系统	优缺点
ode45()	四阶/五阶龙格-库塔	中	非刚性方程	属于单步算法(只需要前一步的解即可计算出当前的解)，不需要附加初始值，因而，计算过程中随意改变步长也不会增加任何计算量。通常 ode45()对很多问题来说都是首选的最好方法
ode23()	二阶/三阶龙格-库塔	低	非刚性方程	属于单步算法，在误差容许范围较宽或者存在轻微刚度时性能比 ode45()好
ode113()	可变阶 Adams PECE 算法	低~高	非刚性方程	属于多步解法(需要前几步的解来计算当前解)。比 ode45()更适合解决误差允许范围比较严格的情况
ode15s()	可变阶的数值微分公式算法（NDFS）	低~中	刚性方程	属于多步解法。如果 ode45()解法速度很慢，很可能系统是刚性的，可以尝试采用该算法

续表

函数名	采用算法	精度	适用系统	优缺点
ode23s()	基于改进的 Rosenbrock 公式	低	刚性方程	属于单部解法。比 ode15s()更适用于误差容许范围较宽的情况。可以解决一些 ode15s()效果不好的刚性方程
ode23t()	自由内插实现的梯形规则	低	轻微刚性方程	适用于轻微刚性系统，给出的解无数值衰减
ode23tb	TR-BDF2 方法，即龙格-库塔公式的第一级采用梯形规则、第二级采用 Gear 法	低	刚性方程	对于误差允许范围比较宽的情况，比 ode15s 效果好

解显式常微分方程的 7 个函数的调用格式是完全一样的，下面以函数 ode45()为例来说明它们的基本调用格式：

- [*t*,*Y*] = ode45(*odefun*,*tspan*,*y*0)，*odefun* 为代表常微分方程的方程组，其格式为 $y' = f(t, y)$，其中 t 是一个标量，y 是一个列向量，y'是与 y 具有相同长度的列向量。*tspan* 可以是两个元素的向量[*t*0 *tf*]，这时函数返回 *t*0 到 *tf* 时间范围内的常微分方程的解；*tspan* 也可以是[*t*0,*t*1,...,*tf*]，这时函数返回在时间[*t*0,*t*1,...,*tf*]上的常微分方程的解。*y*0 是 *y* 具有相同长度的列向量，用于指定初始值；
- [*t*,*Y*] = ode45(*odefun*,*tspan*,*y*0,*options*)，*options* 参数用于设定微分方程解法器的参数。*options* 可以由函数 odeset()来获得。

例如，求非刚性方程 $y'' - \mu(1-y^2)y' + y = 0$，其中 μ 是参数，在本例中 μ =1。首先是要把该二阶常微分方程改写为一阶常微分方程组：

$$y_1' = y_2$$
$$y_2' = \mu(1-y_1^2)y_2 - y_1$$

然后，把该一阶常微分方程组用一个 M 文件形式的函数来表述，代码设置如下：

```
%ivpodefun.m
%常微分方程
function dydt = ivpodefun(t,y)
dydt      = zeros(2,1);
dydt(1) = y(2);
dydt(2) = (1-y(1)^2)*y(2)-y(1);
```

> 注意：表示一个微分方程的函数必须有时间 *t* 这个输入变量，然而在函数内部计算时可以不使用 *t* 这个变量。

最后，用函数 ode45()来解这微分方程，并画图计算结果，代码设置如下：

```
%ode45_example.m
%解常微分方程
[t,y] = ode45(@ivpodefun,[0 20],[2; 0]);
plot(t,y(:,1),'-',t,y(:,2),'--')
```

```
title('常微分方程的解');
xlabel('t');
ylabel('y');
legend('y','y 的一阶导数');
```

由上述语句得到如图 6-28 所示的示意图。

如果要处理含参数的微分方程组可以把参数作为表示微分方程组的函数的第 3 个输入变量。然后再调用函数 ode45()时，把参数值作为最后一个输入变量。其微分方程组可以表示为：

```
%ivpodefun_para.m
%常微分方程
function dydt = ivpodefun_para(t,y,u)
dydt      = zeros(2,1);
dydt(1) = y(2);
dydt(2) = u*(1-y(1)^2)*y(2)-y(1);
```

如果要求参数 u=10 的微分方程组，可以用如下代码实现：

```
[t,y] = ode45(@ivpodefun_para,[0 20],[2; 0],[],10)
```

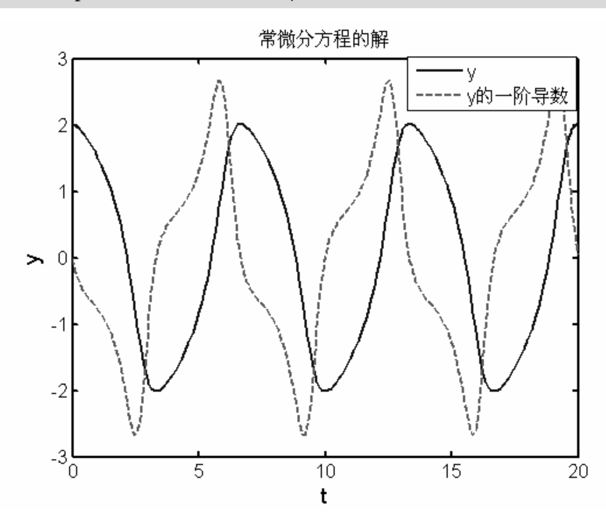


图 6-28　常微分方程解

2．设置常微分方程组解法器参数

正如上小节介绍的，在调用常微分方程组函数时可以输入一个 *options* 结构体，从而设置解法器的参数。该 *options* 结构体可以用函数 odeset()来设定，其调用格式如下：

- *options* = odeset('*name*1',*value*1,'*name*2',*value*2,...)，用参数名和参数值对来设定解法器的参数；
- *options* = odeset(*oldopts*,'*name*1',*value*1,...)，修改原来的解法器 *options* 结构体 *oldopts*，只改变指定的某些参数的值；
- *options* = odeset(*oldopts*,*newopts*)，合并两个解法器 *options* 结构体 *oldopts* 和 *newopts*，这两个结构体中值不同的参数，采用 *newopts* 中的参数值；
- odeset，显示所有的参数值和它们的默认值。

常微分方程组解法器参数如表 6-11 所示。

表 6-11　　常微分方程组解法器参数

参数名	可取值	默认值	用途描述
RelTol	正标量	1e-3	用于所有分量的相对误差，解法器的积分估计误差必须小于相对误差与解的乘积并且小于绝对误差
AbsTol	正标量或者向量	1e-6	绝对误差允许范围，如果是标量该绝对误差应用于所有的分量；如果是向量则单独指定每一个分量的绝对误差
NormControl	'on'或者'off'	'off'	如果该值为'on'，解法器采用积分估计误差的模来控制计算精度；如果该值为'off '，解法器采用更加严格的精度控制策略，即严格控制每一分量的计算精度
OutputFcn	函数句柄	@odeplot 或者[]	每个时间步长计算完后，解法器调用该函数来输出。如果调用 ode 类函数时，没有输出变量，则默认采用 odeplot 来输出数据；如果有输出变量，则默认值是[]，即不输出结果。可选的输出函数有： odeplot：一维时域画图 odephas2：二维相位平面画图 odephas3：三维相位平面画图 odeprint：在命令行输出解
OutputSel	正整数向量	所有分量的下标	OutputSel 向量包含的下标所对应的分量被送给输出函数 OutputFcn 输出。默认情况下，所有分量都输出
Refine	正整数	1 或者 4（ode45）	如果 refine 大于 1，则输出结果被插值，从而提供输出结果的精度
Stats	'on'或者'off'	'off'	如果该值为'on'，输出计算耗费时间，否则不输出
Jacobian	函数或者常数矩阵	无	指定常微分方程的 Jacobian 矩阵
JPattern	稀疏矩阵	无	给出微分方程的 Jacobian 矩阵稀疏样式，如果微分方程的 Jacobian 矩阵的元素不为 0 则 JPattern 相应元素为 1，否则为 0
Vectorized	'on'或者'off'	'off'	ode 函数是否被向量化，如果该值等于 1，则 ode 函数 *x* 形式为[f(*t*,*y*1) f(*t*,*y*2) ...]，否则返回格式为 f(*t*,[*y*1 *y*2 ...])
Events	函数	无	定位事件
Mass	常数矩阵或者函数	无	指定线性隐式常微分方程组的加权函数 $M(t,y)$
MStateDependence	'none' 'weak'或者'strong'	'weak'	说明加权函数 $M(t,y)$是否依赖于 y
MvPattern	稀疏矩阵	无	抖$M(t,y)v)/\ \ y$ 的稀疏矩阵样式。用于加权矩阵 M 与 y 强相关的情况
MassSingular	'yes' 'no'或者'maybe'	'maybe'	加权函数 M(*t*,*y*)是否奇异
InitialSlope	向量	无	初始一阶导数值 *yp*0，满足 M(*t*0,*y*0)**yp*0 = f(*t*0,*y*0)
MaxStep	正标量	自动选择	最大步长值
InitialStep	正标量	自动选择	解法器自动选择初始步长
MaxOrder	1，2，3，4，5	5	ode15s 采用的最大阶数
BDF	'on'或者'off'	'off'	在 ode15s 算法中是否采用 BDF 算法

例如，设置解法器的输出函数为二维相位平面画图，代码设置如下：

```
%odeset_example.m
%解常微分方程
```

```
option = odeset('RelTol',1e-6,'OutputFcn','odephas2');
[t,y] = ode45(@ivpodefun,[0 20],[2; 0],option);
xlabel('y');
ylabel('y 的一阶导数');
```

由上述语句得到如图 6-29 所示的示意图。

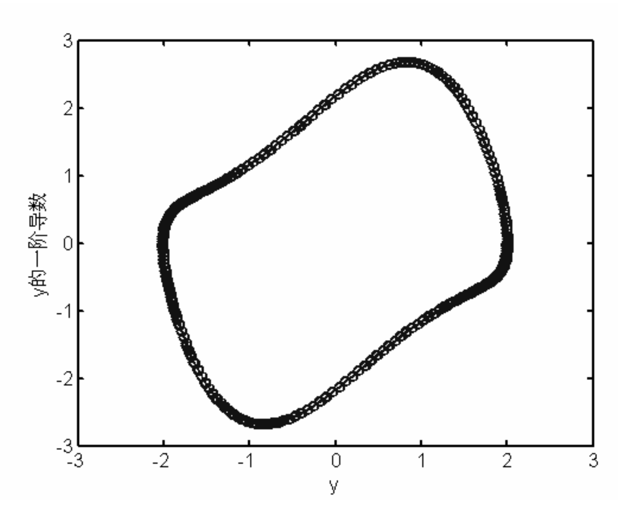


图 6-29　二维相位平面画图

3．线性隐式常微分方程组

线性隐式常微分方程组可以利用 ode 类函数的解法器参数 *options* 来解决。首先创建 M 文件函数来表示加权函数 M(*t*,*y*)，然后用 odeset()函数来把加权函数设置到解法器的 *Mass* 参数中，最后用 ode 类函数来解微分方程组。

例如，求解线性隐式常微分方程组 $(y+1)y' = y^2 + 2y - 2$，首先创建表示微分方程的函数 f(*t*,*y*)和 M(*t*,*y*)，表示 f(*t*,*y*)的函数，代码设置如下：

```
%odefun_LinImp.m
%线性隐式常微分方程
function dydt =   odefun_LinImp(t,y)
dydt = y.^2 + 2*y -2;
```

表示 M(*t*,*y*)的函数，代码设置如下：

```
%odefun_LinImp_mass.m
%线性隐式常微分方程
function mass =   odefun_LinImp_mass(t,y)
mass = y + 1;
```

求解微分方程，代码设置如下：

```
%ode_LinImp_example.m
%解常微分方程
option = odeset('RelTol',1e-6,'OutputFcn','odeplot','Mass',@odefun_LinImp_mass);
```

```
[t,y] = ode45(@odefun_LinImp,[0 2],2,option);
xlabel('t');
ylabel('y');
```

由上述语句得到如图 6-30 所示的示意图。

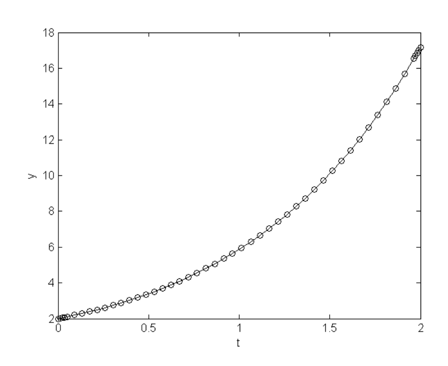


图 6-30　线性隐式常微分方程的解

4．完全隐式常微分方程组

MATLAB 7.0 的新特性之一是提供了一个完全隐式常微分方程组的解法器 ode15i()。其调用格式如下：

- [*t*,*Y*] = ode15i(*odefun*,*tspan*,*y*0,*yp*0)，*odefun* 是代表完全隐式常微分方程组的函数，其形式为 f(*t*,*y*,*y'*)。*tspan* 可以是两个元素的向量[*t*0 *tf*]，这时函数返回 *t*0 到 *tf* 时间范围内的常微分方程的解；*tspan* 也可以是[*t*0,*t*1,...,*tf*]，这时函数返回在时间[*t*0,*t*1,...,*tf*]上的常微分方程的解。*y*0 和 *yp*0 用于指定微分方程的初始值，初始值必须是自洽的，即满足 f(*t*,*y*0,*yp*0)=0；
- [*t*,*Y*] = ode15i(*odefun*,*tspan*,*y0*,*yp0*,*options*)，用 *options* 结构体设定解法器的参数，*options* 结构体可以由函数 odeset()来得到。

处理完全隐式常微分方程组的一个重要问题是如何得到自洽的初始值。MATLAB 7.0 提供函数 decic()来得到自洽的初始值。其调用格式如下：

- [*y*0*mod*,*yp*0*mod*] = decic(*odefun*,*t*0,*y*0,*fixed_y*0,*yp*0,*fixed_yp*0)，*odefun* 是代表完全隐式常微分组的函数。*t*0 值为初始时间，*y*0 和 *yp*0 是猜测的初始值。函数 decic()会通过不断改变初始值，最终得到自洽的初始值。如果希望在改变初始值的过程中保持某一个分量不变，可以通过 *fixed_y*0 输入来实现。当 *fixed_y*0*(i)*=1 时，*y*0*(i)*不会被改变。*fixed_yp*0 的作用与 *fixed_y*0 是一样的；
- [*y*0*mod*,*yp*0*mod*] = decic(*odefun*,*t*0,*y*0,*fixed_y*0,*yp*0,*fixed_yp*0,*options*)，用 *options* 结构体设定解法器的参数，*options* 结构体可以由函数 odeset()来得到。

例如，解完全隐式常微分方程 Weissinger 方程，其形式如下：

$$ty^2(y')^3 - y^3(y')^2 + t(t^2+1)y' - t^2y = 0$$

当初值等于 $\sqrt{3/2}$ 时，方程的解析解为 $y(t) = \sqrt{t^2+0.5}$。本例将用数值解法来求解这

个问题。首先创建一个代表该方程的 M 文件函数，代码设置如下：

```
%ode_weissfun.m
%代表 Weissinger 方程的 M 文件函数
function exp = ode_weissfun(t,y,dydt)
exp = t*y.^2*dydt.^3-y.^3*dydt.^2+t*(t^2+1)*dydt-t^2*y;
```

解微分方程的程序，代码设置如下：

```
%ode15i_example.m
%解完全隐式微分方程
t0 = 1;                                                    %猜想的初值
y0 = sqrt(3/2);
yp0 = 0;
[y0,yp0] = decic(@ode_weissfun,t0,y0,1,yp0,0);           %求出自洽初值，并保值 y0 值不变
[t,y] = ode15i(@ode_weissfun,[1 10],y0,yp0);             %求在[1 10]区间内的解
ytrue = sqrt(t.^2 + 0.5);                                  %解析解
plot(t,y,t,ytrue,'o');                                     %画图比较数值解与解析解
xlabel('t');
ylabel('y');                                               %
```

由上述语句得到如图 6-31 所示的示意图。

6.5.2 延迟微分方程组数值解

延迟微分组方程的形式如下：

$$y'(t) = f(t, y(t), y(t-t_1), \cdots, y(t-t_k))$$

在 MATLAB 7.0 中使用函数 dde23()来解延迟微分方程。其调用格式如下：

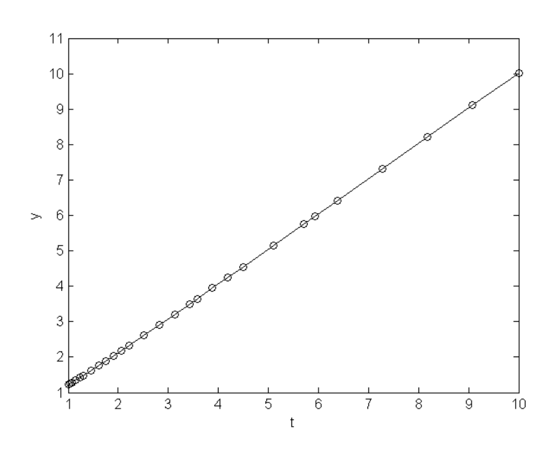


图 6-31　完全隐式常微分方程解

- *sol* = dde23(*ddefun*,*lags*,*history*,*tspan*)，其中 *ddefun* 是代表延迟微分方程的 M 文件函数，*ddefun* 的格式为 *dydt* = ddefun(*t*,*y*,*Z*)，*t* 是当前时间值，*y* 是列向量，*Z*(:,j)代表 $y(t-t_k)$，

而 t_k 值在第二个输入变量 lags(*k*)中存储。*history* 为 *y* 在时间 *t*0 之前的值，可以有 3 种方式来指定 *history*。第 1 种是用一个函数 *y*(*t*)来指定 *y* 在时间 *t*0 之前的值；第 2 种方法是用一个常数向量来指定 *y* 在时间 *t*0 之前的值，这时 *y* 在时间 *t*0 之前的值被认为是常量；第 3 种以前一时刻的方程解 *sol* 来指定时间 *t*0 之前的值。*tspan* 是两个元素的向量[*t*0 *tf*]，这时函数返回 *t*0~*tf* 时间范围内的延迟微分方程组的解；

- *sol* = dde23(*ddefun*,*lags*,*history*,*tspan*,*option*)，*option* 结构体用于设置解法器的参数，*option* 结构体可以由函数 ddeset()来获得。

函数 dde23()的返回值是一个结构体，它有 7 个属性，其中重要的属性有如下 5 个：

- *sol.x*，dde23 选择计算的时间点；
- *sol.y*，在时间点 *x* 上的解 *y*(*x*)；
- *sol.yp*，在时间点 *x* 上的解的一阶导数 *y'*(*x*)；
- *sol.history*，方程初始值；
- *sol.solver*，解法器的名字'dde23'。

其他两个属性为 sol.stat 和 sol. discont。

如果需要得到在[*t*0　*tf*]之间 *tint* 时刻的解，可以使用函数 deval，其用法为：*yint* = deval(*sol*,*tint*)，*yint* 是在 *tint* 时刻的解。

例如，求解如下延迟微分方程组：

$$y'_1 = y_1^2(t-3) + y_2^2(t-1)$$
$$y'_2 = y_1(t) + y_2(t-1)$$

初始值为：

$$y_1(t) = 1$$
$$y_2(t) = t-2 \qquad (t<0)$$

首先，确定延迟向量 lags，在本例中 lags=[1 3]。

其次，创建一个 M 文件形式的函数表示延迟微分方程组，代码设置如下：

```
%ddefun.m
%延迟微分方程
function dydt=ddefun(t,y,Z)
dydt = zeros(2,1);
dydt(1) = Z(1,2).^2 + Z(2,1).^2;
dydt(2) = y(1) + Z(2,1);
```

然后，创建一个 M 文件形式的函数表示延迟微分方程组的初始值，代码设置如下：

```
%ddefun_history.m
%延迟微分方程的历史函数
function y=ddefun_history(t)
y = zeros(2,1);
y(1) = 1;
y(2) = t-2;
```

最后，用 dde23 解延迟微分方程组和用图形显示解，代码设置如下：

```
%dde_example.m
%解延迟微分方程的例子
lags = [1 3];                                          %延迟向量
sol = dde23(@ddefun,lags,@ddefun_history,[0,1]);       %解方程
hold on;
plot(sol.x,sol.y(1,:),'b-');                           %画出结果
plot(sol.x,sol.y(2,:),'r-.');
title('延迟微分方程的解');
xlabel('t');
ylabel('y');
legend('y_1','y_2',2);
```

由上述语句得到如图 6-32 所示的示意图。

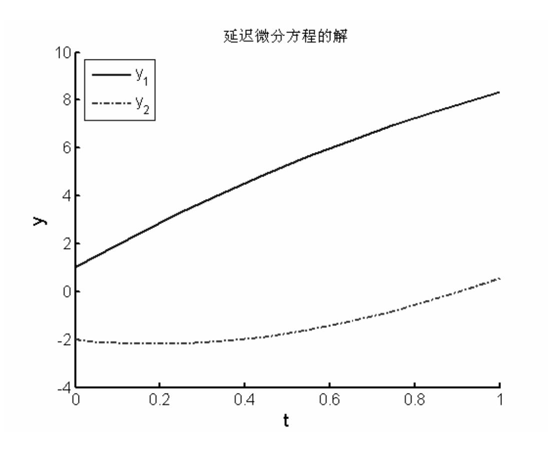


图 6-32　延迟微分方程组的解

6.5.3　常微分方程组的边界问题

常微分方程组的边界问题的形式如下：

$$\frac{dy}{dx} = f(x, y)$$

同时要指定函数 $y(x)$在某些边界条件下的值，然后在边界条件下求函数 $y(x)$，其中 x 是独立变量。MATLAB 7.0 中使用函数 bvp4c()来处理常微分方程组的边界问题。该函数解决第一类边界条件问题，即边界条件是：

$$g(y(a), y(b)) = 0$$

其中 a 和 b 是求解区间的下界和上界，即要求解出 y 在区间$[a\ b]$上的值。

常微分方程组的边界问题与常微分方程组的初值问题的不同之处在于：常微分方程组的初值问题总是有解的，然而常微分方程组的边界问题有时会出现无解的情况，有时会出现有限个解，有时会出现无穷多个解。因此在解常微分方程组的边界问题时，不可或缺的一部分

工作是提供猜测解。猜测解决定了解边界问题的算法性能，甚至算法是否成功。

在边界问题中，还经常出现附加的未知参数，其方程形式如下：

$$\frac{dy}{dx} = f(x, y, p)$$

其边界条件为：

$$g(y(a), y(b), p) = 0$$

在这种情况下，边界条件必须充分，从而能够决定未知参数 p。

函数 bvp4c()的调用格式如下：

- *sol* = bvp4c(*odefun*,*bcfun*,*solinit*)，*odefun* 是描述常微分方程组的函数，其格式为 *dydx* = odefun(x,*y*)，或者包含未知参数 *dydx* = odefun(*x*,*y*,*parameters*)。*bcfun* 是描述边界条件的函数，其格式为 *res* = bcfun(*ya*,*yb*)，或者包含未知参数 *res* = bcfun(*ya*,*yb*,*parameters*)。*solinit* 是对方程解的猜测解，它是一个结构体，包含 *x*、*y* 和 *parameters*3 种属性。*solinit* 必须满足 *solinit*.*x*(1)=*a* 和 *solint*.*x*(*end*)=*b*，*parameters* 是对未知参数的一个猜测解。*solinit* 可以由函数 *bvpinit*()得到；
- *sol* = bvp4c(*odefun*,*bcfun*,*solinit*,*options*)，使用 *options* 结构体来设定解法器的参数。*options* 结构体可以由函数 bvpset()得到。

边界问题的猜测解可以由函数 solinit()得到，其调用格式如下：

- *solinit* = bvpinit(*x*,*yinit*)，*x* 是猜测解的自变量 *x* 取值，*yinit* 猜测解的因变量 *y* 取值。如果需要在区间[*a b*]上求解方程，*x* 取 linspace(*a*,*b*,10)就足够了，在解变化比较快的时候，需要用长的自变量 *x*；
- *solinit* = bvpinit(*x*,*yinit*,*parameters*)，*parameters* 未知参数的猜测解。

例如，求 Mathieu 方程的特征值。Mathieu 方程的形式为：

$$y'' + (l - 2q\cos 2x)y = 0$$

其中 *q*-1 是 Mathieu 方程的阶数，为已知参数。λ 是未知参数 Mathieu 方程的特征值。为在本例中 *q*=5，即求 4 阶 Mathieu 方程的特征值 λ。首先把方程改写成一阶常微分方程的形式：

$$y'_1 = y_2$$
$$y'_2 = -(l - 2q\cos 2x)y_1$$

指定边界条件为：

$$y'(0) = 0$$
$$y'(p) = 0$$
$$y(0) = 1$$

其次，创建 M 文件来描述 Mathieu 方程，代码设置如下：

```
%bvp_Mathieufun.m
%4 阶 Mathieu 方程
function dydx=bvp_Mathieufun(x,y,lambda)
```

```
q = 5;
dydx = zeros(2,1);
dydx(1) = y(2);
dydx(2) = -(lambda - 2*q*cos(2*x))*y(1);
```

然后，创建 M 文件来描述方程的初值，代码设置如下：

```
%Mathieu_initfun.m
%4 阶 Mathieu 方程的初始值
function yinit = Mathieu_initfun(x)
yinit(1) =    cos(4*x);
yinit(2) =    -4*sin(4*x);
```

还需要，创建 M 文件来描述方程的边界条件，代码设置如下：

```
%Mathieu_bcfun.m
%4 阶 Mathieu 方程的边界条件
function res = Mathieu_bcfun(ya,yb,lambda)
res = zeros(3,1);
res(1) =    ya(2);
res(2) =    yb(2);
res(3) =    ya(1)-1;
```

最后，用函数 bvp4c 来解方程，并画图显示结果，代码设置如下：

```
%bvp4c_example.m
%Mathieu 方程在边界条件下的解
lambda = 15;                                                          %未知参数猜测解
solinit = bvpinit(linspace(0,pi,10),@Mathieu_initfun,lambda);%求初始值
sol = bvp4c(@bvp_Mathieufun,@Mathieu_bcfun,solinit);                 %求解方程
xint = linspace(0,pi);                                                %画图
Sxint = deval(sol,xint);
plot(xint,Sxint(1,:))
axis([0 pi -1 1.1])
title('Mathieu 方程在边界条件下的解');
xlabel('x')
ylabel('y')
text(0.1,1,['4 阶 Mathieu 方程的特征值  = ' num2str(sol.parameters)]);
```

由上述语句得到如图 6-33 所示的示意图。

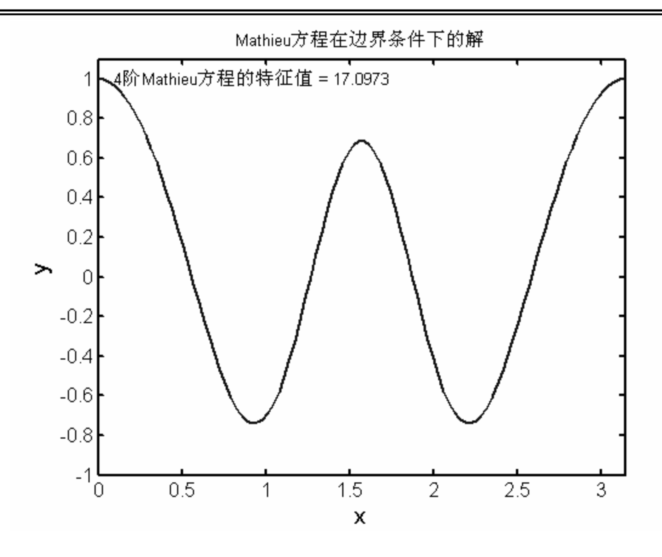


图 6-33　Mathieu **方程在边界条件下的解**

第 7 章　Simulink 仿真环境

通过本章的学习，读者将获得以下知识：

- 理解 Simulink 的概念及其应用；
- 掌握如何使用 Simulink 搭建系统模型及其特点；
- 掌握如何使用 Simulink 进行系统仿真并进行调试。

在学习本章时，读者既可按照作者介绍的先后顺序去逐步深入了解，也可以先通过第 7.2 节或第 7.11 节的学习来掌握 Simulink 建模的大概过程和基本步骤，然后再通过学习其他小节的内容来掌握建模过程中的各种具体的操作细节和一些仿真计算的理论问题。

7.1　Simulink 概述

下面先向读者介绍一个非常简单的示例，旨在使读者在进行深入学习之前先对 Simulink 有一个感性的认识。

【创建模型目的】：计算两个不同频率正弦函数相加后再积分的结果，并显示结果的波形。

【创建模型的步骤】：

- 在 MATLAB7.0 界面窗口选择 File→New→Model 菜单项，弹出如图 7-1 和图 7-2 所示的窗口；
- 在如图 7-1 所示窗口左边选中 Sources 库，然后在右边选 Sine Wave 模块并按住鼠标左键不放，将它拖到如图 7-2 所示的窗口中。重复该操作添加第二个 Sine Wave 模块；

按上一步中的方法向图 7-2 的窗口中添加 Math Operations 库中的 Add 模块、Continuous 库中的 Integrator 模块和 Sinks 库中的 Scope 模块；

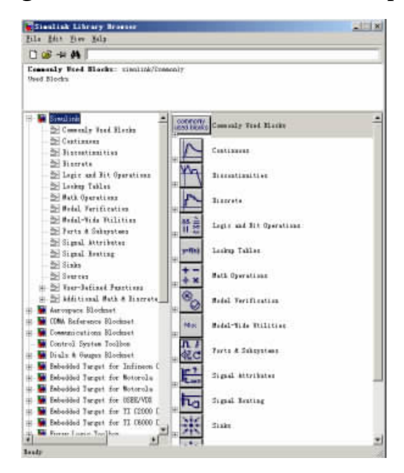


图 7-1　Simulink 模块库浏览器

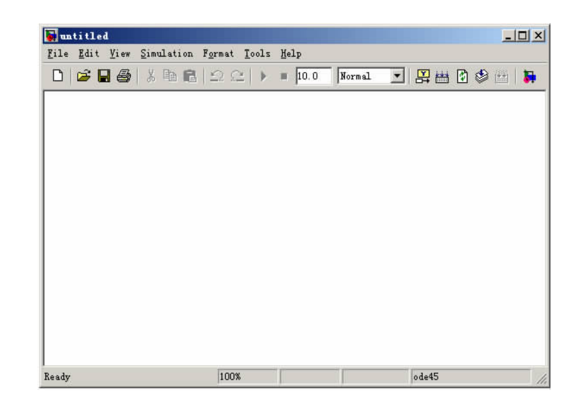


图 7-2　新建模型窗口

- 按如图 7-3 所示连接模块。连接模块的操作方法：用鼠标指向源模块的输出端口，当鼠标变成十字架形时按住鼠标左键不放，然后拖动鼠标指向目标模块输入端口后松开；
- 设置 Sine Wave 模块的参数。鼠标左键双击 Sine Wave 模块，弹出如图 7-4 所示参数设置对话框，设置 Frequency 为 2，然后单击“OK”按钮；
- 单击▸按钮运行仿真，然后用鼠标左键双击模型中 Scope 模块，弹出如图 7-5 所示的输出波形。

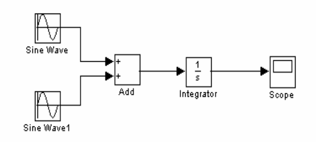


图 7-3　简单示例模型

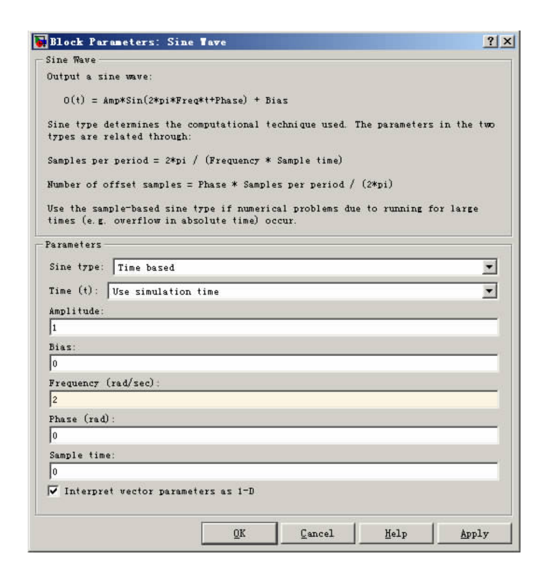


图 7-4　设置参数

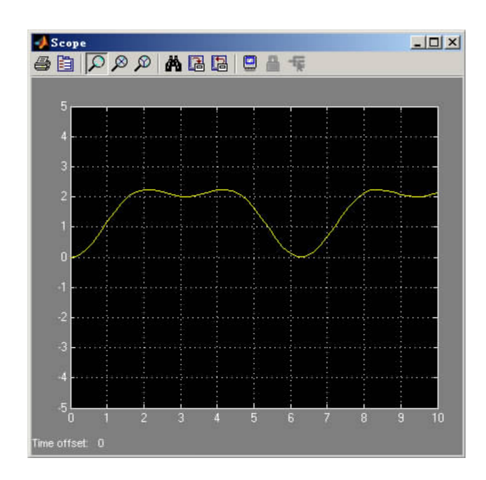


图 7-5　输出波形

7.1.1 Simulink 的概念

Simulink 是 MATLAB 提供的实现动态系统建模和仿真的一个软件包，它是 MATLAB 的一个重要组成部分，而且具有相对独立的功能和使用方法。它支持线性和非线性、连续时间系统、离散时间系统、连续和离散混合系统建模，且系统可以是多进程的。

Simulink 的一个突出特点是它支持图形用户界面（GUI），模型由模块组成的框图来表示。用户建模通过简单的单击和拖动鼠标的动作就能完成。如果把建模比做建造房子，那么采用高级语言或 MATLAB 语言直接编写仿真程序的方式来构建模型就好比是从一堆沙子开始来建房子，这种方式不但麻烦容易出错，而且有许多重复操作，建造者的主要精力不是放在房子的结构设计上，而是浪费在如何将沙子变成砖块以及如何将它们组合起来等技术性的问题上，这显然不利于设计者去设计出更多有创造性的结构。在计算机仿真里，就等于设计者把精力放在具体算法的实现上，而不是模型和算法设计本身。Simulink 通过自带的模块库（如图 7-1 所示）为用户提供多种多样的基本功能模块，用户可以直接调用这些模块，而不必从最基本的做起。这样就可以让设计者把精力放在更为重要的更具创造性的算法和模块结构的设计上来。

Simulink 的每个模块对于用户来说都相当于一个“黑匣子”，用户只需知道模块的输入和输出以及模块功能即可，而不必管模块内部是怎么实现的。因此，用户使用 Simulink 进行系统建模的任务就是如何选择合适的模块并把他们按照自己的模型结构连接起来，最后进行调试和仿真。如果仿真结果不满足要求，可以改变模块的相关参数再运行，直到结果满足要求为止。至于在仿真时各个模块是如何执行的、各模块间是如何通信的、仿真的时间是如何采样的以及事件是如何驱动的等细节问题，用户都不用去管，因为这些事情 Simulink 都解决了。如何添加和删除模块、如何连接各个模块以及如何修改模块的参数和属性等问题将在本章后面的各小节会陆续给予详细的介绍。

Simulink 最新版本是 Simulink6.0（包含在 MATLAB7.0 里），启动 Simulink 有 3 种方式。

- 在 MATLAB 7.0 的命令窗口直接键入“>>Simulink”命令；
- 用鼠标左键单击 MATLAB 7.0 工具条上的 Simulink 按钮；
- 在 MATLAB 7.0 菜单上选择“File→New→Model”选项。

运行后会弹出如图 7-1 所示的 Simulink 模块库浏览器窗口，使用第 3 种方式打开时还会弹出如图 7-2 所示的新建模型窗口。单击图 7-1 工具条左边的图标（建立新模型）也会弹出如图 7-2 所示的窗口。

和 Windows 窗口类似，在 Simulink 的模块窗口和模块库窗口的“View”菜单下选择或取消“Toolbar”和“Status Bar”选项，就可以显示或隐藏工具条和状态条。在进行仿真过程中，模型窗口的状态条会显示仿真状态、仿真进度和仿真时间等相关信息。

> 注意：本章是以 Simulink 6.0 版为例进行介绍的。

7.1.2 Simulink 的工作环境

按照第 7.1.1 节介绍的方法启动 Simulink 后，就可以看到如图 7-1 所示的 Simulink 模块库浏览器。在建模的过程中会经常对它进行操作，下面介绍该界面各个部分的用途，如图 7-6

所示。

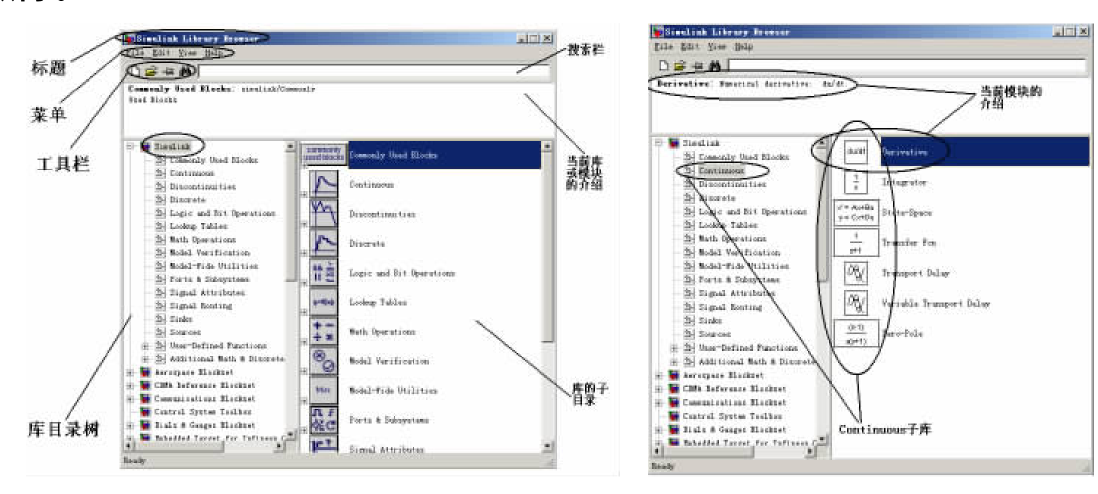


图 7-6　Simulink 模块库浏览器的结构

对于熟悉 Windows 窗口的读者来说，Simulink 模块库浏览器的界面应该很容易掌握，在这里就不做详细介绍了。读者可以单击每个标题前的加号，将列出库里的模块，而且还能了解到每个模块的功能说明，在第 7.1.6 小节将向读者详细地介绍其中一些库中的常用模块。

接下来向读者介绍 Simulink 模型窗口界面。模型建立好之后开始进行各项操作，大部分的操作都是在模型窗口完成的，因此读者需要熟练掌握这些操作，并且要了解其中的各个菜单（如表 7-1~表 7-6 所示）和按钮（如图 7-11 所示）的功能。

1．“ File ”菜单

“ File ”菜单中各选项的名称与功能如表 7-1 所示。

表 7-1　“ File ”菜单

主要子菜单	功能
New	新建模型（Model）或库（Library）
Open	打开一个模型
Close	关闭模型
Save	保存模型
Save as	另存为
Model Properties	打开“ 模型属性 ”对话框
Preferences	打开“ 模型参数设置 ”对话框（如图 7-7 所示），“ Preferences ”对话框主要用于设置一些用户界面的显示形式，如颜色、字体等
Source control	设置 Simulink 与 SCS 的接口
Print	打印模型或模块图标到一个文件
Print Details	生成 HTML 格式的模型报告文件，包括模块的图标和模块参数的设置等，如图 7-8 所示
Print Setup	打印模型或模块图标
Exit MATLAB	退出 MATLAB 7.0

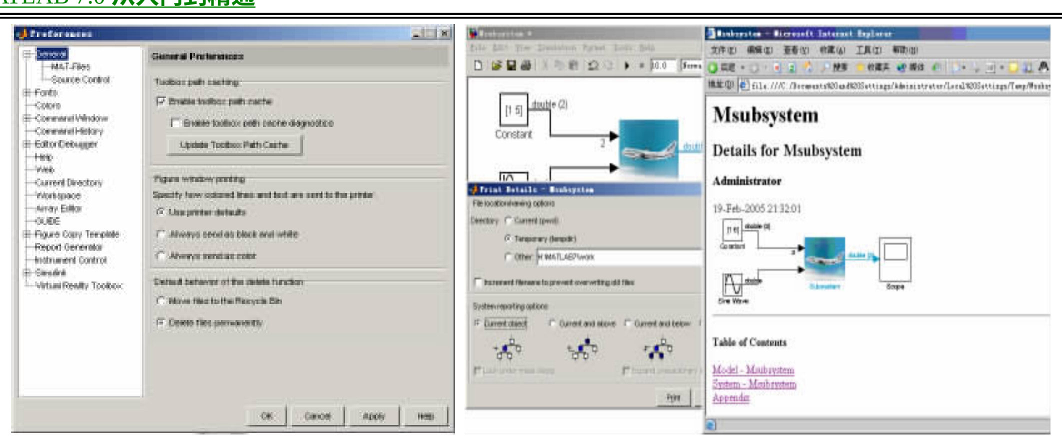


图 7-7 “Preferences”对话框　　　　图 7-8 模型报告文件

2．“Edit”菜单

“Edit”菜单中各选项的名称与功能如表 7-2 所示。

表 7-2　“Edit”菜单

主要子菜单	功能
Copy Model to Clipboard	把模型当图片拷贝下来
Explore	打开模型浏览器，当有模块被选中时才可用
Block Properties	打开模块属性对话框，当有模块被选中时才可用
<Blockname> Parameters	打开模块参数设置对话框，当有模块被选中时才可用
Create Subsystem	创建子系统，当有模块被选中时才可用
Mask Subsystem	封装子系统，当有子系统被选中时才可用
Look under Mask	查看子系统内部构成，当有子系统被选中时才可用
Signal Properties	设置信号属性，当有信号被选中时才可用
Edit Mask	编辑封装，当有子系统被选中时才可用
Subsystem Parameters	打开子系统参数设置对话框，当有子系统被选中时才可用
Mask Parameters	封装好的子系统的参数设置，当有被封装过的子系统被选中时才可用

3．“View”菜单

该菜单的内容较多，但大部分命令都很容易理解掌握。其中“Model Browser Options”、“Block Data Tips Optioons”和“Port Values”用于设置在鼠标指针移到某一模块时有没有模块及其端口的相关提示信息，菜单的子选项用于选择在提示信息中显示哪些内容。“View”菜单中各主要选项具体的名称与功能如表 7-3 所示。

表 7-3　“View”菜单

主要子菜单	功能
Block Data Tips Optioons	用于设定在鼠标指针移到某一模块时是否显示模块的相关提示信息（如模块名、模块参数名及其值和用户定义描述字符串）
Library Browser	打开如图 7-1 所示的模型库浏览器
Port Values	设置如何通过鼠标操作来显示模块端口的当前值
Model Explorer	打开如图 7-9 所示的模型资源管理器，将模块的参数设置、仿真参数设置以及解法器选择、模块的各种信息等集成到一个界面来设置

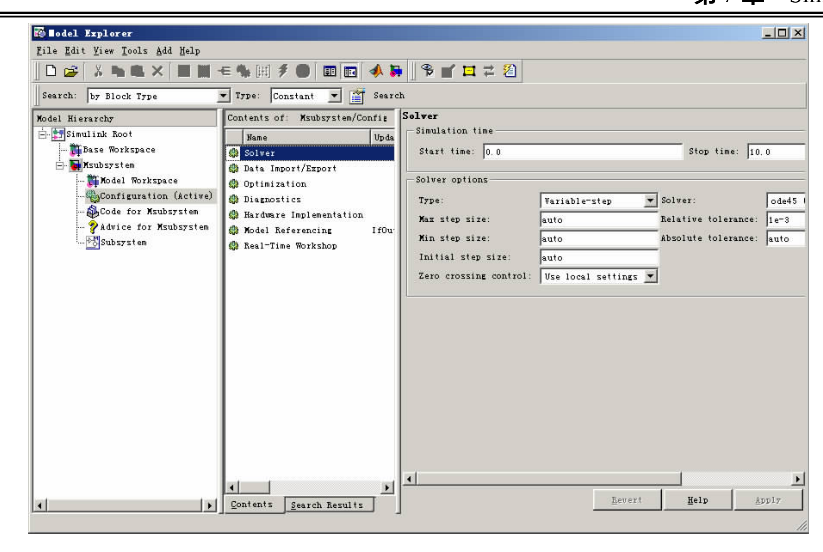


图 7-9　模型资源管理器

4．“ Simulation ”菜单

“ Simulation ”菜单如图 7-10 所示，下面将介绍该菜单中各选项的功能。

“ Start ”：开始运行仿真。

“ Stop ”：停止仿真。

“ Configuration Parameters ”：设置仿真参数和选择解法器（设置方法在第 7.7.2 节介绍）。“ Normal ”、“ Accelerator ”、“ External ”分别表示正常工作模式、加速仿真和外部工作模式。

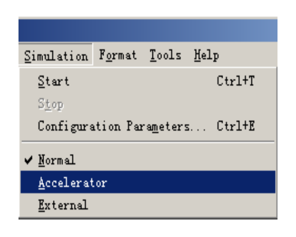


图 7-10　“ Simulation ”菜单

5．“ Format ”菜单

该菜单主要用于设置字体、屏幕颜色、模块名的显示、模块显示颜色、信号和端口类型和宽度等，各项都较容易掌握，各子菜单的功能如表 7-4 所示。

表 7-4　“ Format ”菜单

主要子菜单	功能
Flip Name	翻转模块名字，当有模块被选中时才可用
Flip Block	翻转模块图标，当有模块被选中时才可用
Rotate Block	旋转模块图标，当有模块被选中时才可用
Show Drop Shadow	给模块添加阴影，当有模块被选中时才可用
Port/Signal Displays	显示端口和信号的相关信息，其中“ Sample Time Colors ”选项根据模块的采样时间来设置不同的显示颜色
Block Displays	显示模块相关信息，其中“ Sorted Order ”选项显示模块的优先级

6．“ Tools ” 菜单

该菜单的内容较多，大部分是用来打开各种编辑器或设置对话框，用以设置与模型的仿真相关的参数。“ Tools ” 菜单中各主要选项的功能如表 7-5 所示。

表 7-5　“ Tools ” 菜单

主要子菜单	功能
Simulink Debugger	打开调试器（关于调试器的使用在第 7.9 介绍）
Fixed-Point Settings	打开定点设置对话框
Model Advisor	打开模型分析器对话框，帮助用户检查和分析模型的配置
Lookup Table Editor	打开查表编辑器，帮助用户检查和修改模型中的 lookup table (LUT) 模块的参数
Data Class Designer	打开数据类设计器，帮助用户创建 Simulink 类的子类，即创建自定义数据类
Bus Editor	打开 Bus 编辑器，帮助用户修改模型中 Bus 类型对象的属性
Profiler	选中此菜单后，当仿真运行结束后会自动生成并弹出一个仿真报告文件（HTML 格式）
Coverage Settings	打开 “ Coverage Settings ” 对话框，用户可以通过该对话框设置在仿真结束后给出仿真过程中有关 coverage data 的一个 HTML 格式报告文件
Signal & Scope Manager	打开信号和示波器的管理器，帮助用户创建各种类型的信号生成模块和示波器模块
Real-Time WorkShop	可用于将模块转换为实时可执行的 C 代码
External Mode Control Panel	打开外部模式控制板，用于设置外部模式的各种特性
Control Design	用于打开 “ Control and Estimation Tools Manager ” 和 “ Simulink Model Discretizer ” 对话框
Parameter Estimation	用于打开“ Control and Estimation Tools Manager ”窗口，如图 7-11 所示，可用于模型的参数分析
Report Generator	用于打开报告生成器

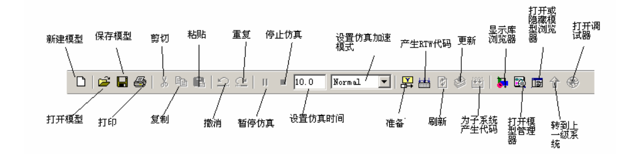


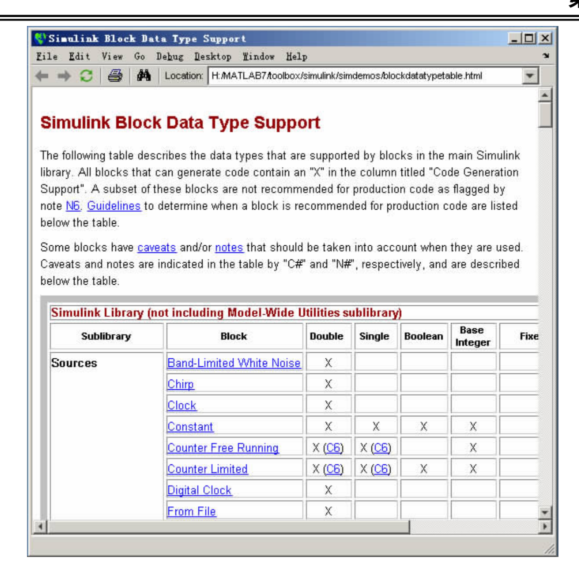
图 7-11　模型窗口的工具栏

7．“ Help ” 菜单

“ Help ” 菜单中各主要选项的名称与功能如表 7-6 所示。

表 7-6　“ Help ” 菜单

主要子菜单	功能
Using Simulink	打开 MATLAB 7.0 的帮助，当前显示在 Simulink 帮助部分
Blocks	打开 MATLAB 7.0 的帮助，当前显示在按字母排序的 Blocks 帮助部分
Blocksets	打开按应用方向分类的帮助
Block Support Table	打开模型所支持的数据类型帮助文件，如图 7-12 所示
Shortcuts	打开 MATLAB 7.0 的帮助，当前显示在鼠标和键盘快捷键设置的帮助部分
S-Function	打开 MATLAB 7.0 的帮助，当前显示在 S-函数的帮助部分
Demos	打开 MATLAB 7.0 的帮助，当前显示在 Demos 页的帮助部分，通过它可以打开许多有用的演示示例
About Simulink	显示 Simulink 的版本



Simulink Block Data Type Support

The following table describes the data types that are supported by blocks in the main Simulink library. All blocks that can generate code contain an "X" in the column titled "Code Generation Support". A subset of these blocks are not recommended for production code as flagged by note N6. Guidelines to determine when a block is recommended for production code are listed below the table.

Some blocks have caveats and/or notes that should be taken into account when they are used. Caveats and notes are indicated in the table by "C#" and "N#", respectively, and are described below the table.

Simulink Library (not including Model-Wide Utilities sublibrary)

Sublibrary	Block	Double	Single	Boolean	Base Integer	Fixe
Sources	Band-Limited White Noise	X				
	Chirp	X				
	Clock	X				
	Constant	X	X	X	X	
	Counter Free Running	X (C6)	X (C6)		X	
	Counter Limited	X (C6)	X (C6)	X	X	
	Digital Clock	X				
	From File	X				

图 7-12　模块支持的数据类型

7.1.3　Simulink 的工作原理

虽然 Simulink 为用户屏蔽了许多繁琐的编程工作，但用户要想更灵活高效地使用它，就必须了解它的工作原理。由前面的示例可以看出 Simulink 建模大体可分成两步。创建模型的图标和控制 Simulink 对它进行仿真。但是用户可能并不明白这些图形化的模型和现实系统之间到底存在着怎样的映射关系，以及 Simulink 是如何对这些模型进行仿真的。下面就来解决这两个问题。

1．图形化的模型和现实系统之间的映射关系

现实中的每一个系统都有输入、输出和状态 3 个基本元素，以及它们之间随时间变化的数学函数关系。Simulink 模型中每一个图形化的模块代表现实系统中一个元件的输入、输出和状态间随时间变化的函数关系，即元件或系统的数学模型，如图 7-13 所示。系统的数学模型是由一系列数学方程式（代数环、微分和差分等式）来描述的，每一个模块都代表一组数学方程，Simulink 中称这些方程为模块或模型的方法（即一组 MATLAB 函数）。模块和模块之间的连线代表系统中各元件输入/输出信号的连接关系，也代表了随时间变化的信号值。



图 7-13　模块的图形化表示

2．Simulink 是如何对图形化模型进行仿真的

Simulink 对模型进行仿真的过程，就是在用户定义的时间段内根据模型提供的信息计算系统的状态和输出的过程。当用户开始仿真，Simulink 分以下几个阶段来运行仿真。

（1）模型编译阶段

Simulink 引擎调用模型编译器，将模型编译成可执行的形式。编译器主要完成以下任务：

- 评价模块参数的表达式以确定它们的值；
- 确定信号属性（如名字、数据类型等）；
- 传递信号属性以确定未定义信号的属性；
- 优化模块；
- 展平模型的继承关系（如子系统）；
- 确定模块运行的优先级；
- 确定模块的采样时间。

（2）连接阶段

Simulink 引擎创建按执行的次序排列的方法（也即函数）运行列表，同时定位和初始化存储每个模块的运行信息的存储器。

（3）仿真环阶段

Simulink 引擎在从仿真的开始时间到结束时间，在每一个时间间隔点按顺序计算系统的状态和输出。这些计算状态和输出的时间点就称作时间步，相邻两个时间点间的长度就称作步长，步长的大小取决于求解器的类型。该阶段又分成两个子阶段。

- 仿真环初始化阶段。该阶段只运行一次，用于初始化系统的状态和输出。
- 仿真环迭代阶段。该阶段在定义的时间段内每隔一个时间步就重复运行一次，用于在每一个时间步计算模型的新的输入、输出和状态，并更新模型使之能反应系统最新的计算值。在仿真结束时，模型能反应系统最终的输入、输出和状态值。

7.1.4 Simulink 模型的特点

使用 Simulink 建立的模型具有以下 3 个特点：

- 仿真结果的可视化；
- 模型的层次性；
- 可封装子系统。

下面通过一个 Simulink 提供的演示示例来说明上述特点。用户先通过菜单命令 Help→MATLAB Help 或是“Help”命令打开如图 7-14 所示的帮助窗口，然后在“Demos”选项卡中选择 Simulink→General Applications→Thermodynamic Model of a House，单击右边文本框中有下画线的文字“Open this model”打开如图 7-15 所示的窗口。用户还可以通过在 MATLAB 7.0 命令窗口输入“>>thermo”命令打开该窗口。

- 单击“开始”按钮，可以看到如图 7-16 所示的仿真结果；
- 鼠标左键双击模型图标中的 House 模块，弹出如图 7-17 所示 House 子系统图标。

对于上述的特点，读者在学完后续章节后将会有更加深刻的理解。

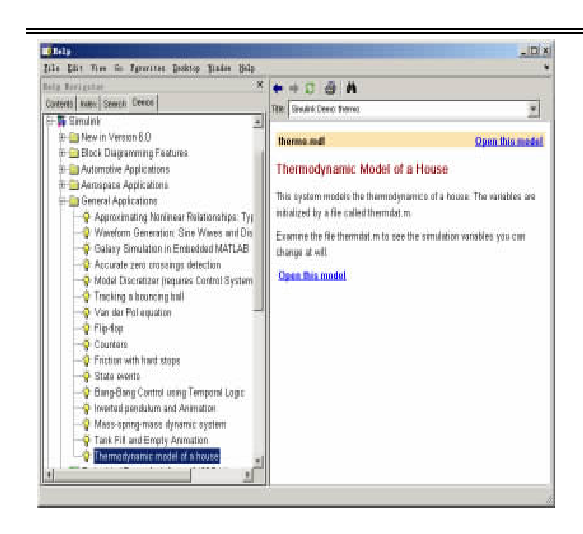


图 7-14　MATLAB 7.0 帮助

图 7-15　thermo 演示模型

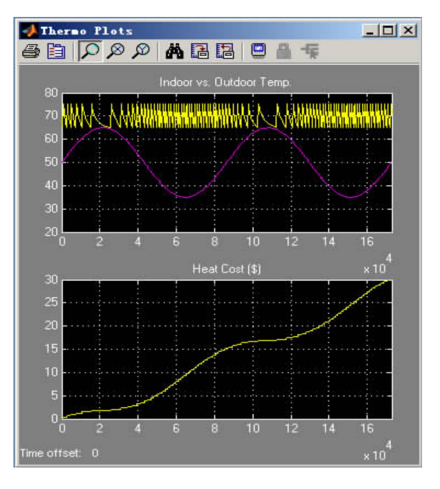


图 7-16　仿真结果可视化

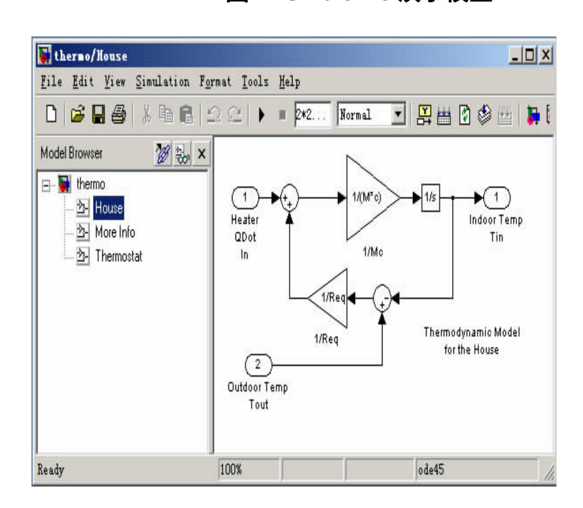


图 7-17　House 子系统图标

7.1.5　Simulink 里的数据类型

在计算机编程语言中，数据类型决定了分配给一个数据的存储资源，决定数据表示的精度、动态范围、性能和存储资源。MATLAB 语言也是一种编程语言，因此 Simulink 中也允许用户说明 Simulink 模型中信号和模块参数的数据类型。

Simulink 在开始仿真之前以及仿真过程中会进行一个额外的检查（系统自动进行，不需手动设置），以确认模型的类型安全性。所谓模型的类型安全性，是指保证该模型产生的代码不会出现上溢或下溢，不至于产生不精确的运行结果。使用 Simulink 默认数据类型（double）的模型都是固有类型安全的。

1．Simulink 支持的数据类型

Simulink 支持所有的 MATLAB 内置数据类型，内置数据类型是指 MATLAB 自己定义的数据类型。表 7-7 列出了所有的 MATLAB 内置数据类型。

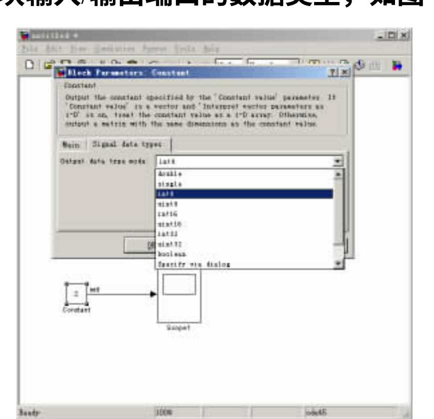
表 7-7　　MATLAB 7.0 内置数据类型

数据类型	类型说明
Double	双精度浮点类型
Single	单精度浮点类型
Int8	有符号 8 位整数
Uint8	无符号 8 位整数
Int16	有符号 16 位整数
Uint16	无符号 16 位整数
Int32	有符号 32 位整数
Uint32	无符号 32 位整数

除了内置数据类型，Simulink 还定义了布尔类型，取值为 0 和 1，它们内部的表示是 uint8（无符号 8 位整数）。所有的 Simulink 模块都默认为 double 类型的数据，但有些模块需要布尔类型的输入，而另外一些模块支持多数据类型输入，还有些支持复数信号。

关于模块的输入/输出信号支持的数据类型的详细说明，用户可以通过模块参数设置对话框（双击模块图标就会弹出对话框）来查看。如果在一个模块的参数设置对话框中没有说明它支持的数据类型的选项，那就表示它只支持 double 类型的数据，如图 7-18 所示。用户还可以选择 Simulink 模型窗口中的“Help”菜单中的“Block Support Table”子菜单，打开如图 7-12 所示的帮助窗口，其中总结了所有 Simulink 库中的模块所支持的数据类型的情况。

选择模型窗口的 Format→Port/Signal Displays→Port Data Types 可以查看信号的数据类型和模块输入/输出端口的数据类型，如图 7-19 所示。



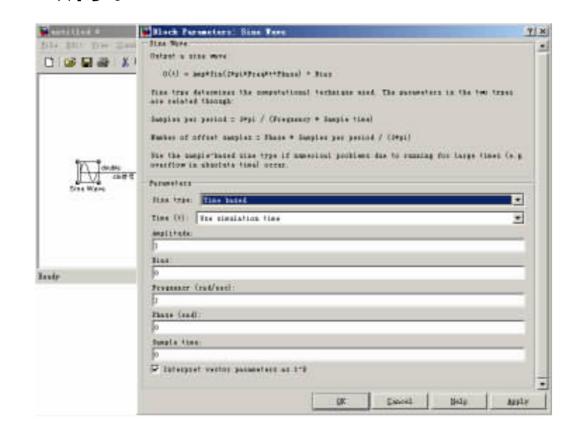


图 7-18　查看模块支持的数据类型

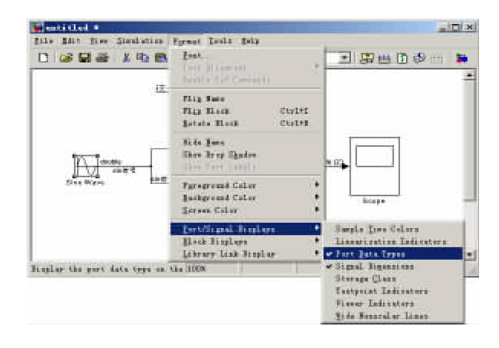


图 7-19　查看信号的数据类型

在设置模块参数时，指定一个值为某一数据类型的方法为 type（value）。例如要把常数模块的参数设为 1.0 单精度表示，则可以在参数对话框输入 single（1.0）。如果模块不支持所设置的数据类型，那么就会弹出错误警告，如图 7-20 所示。

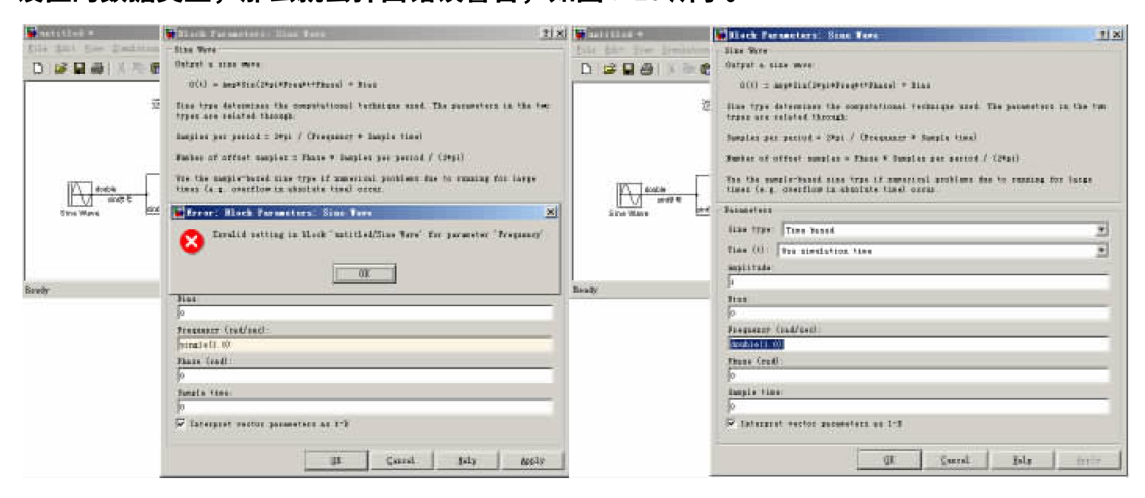


图 7-20　设置模块参数的数据类型

在建立模型时，常常需要向模型提供某种特定类型的输入信号数据，把一个具有特定数据类型的信号引入模型具体有以下几种方法。

- 通过根目录 import 或者 FromWorkspace 模块将 MATLAB 7.0 工作空间中具有指定数据类型的信号数据导入你的模型；
- 建立一个常数模块并设置它的参数为指定的类型；
- 使用 DataTypeConversion 模块将一个信号转换为指定类型的信号。

2．数据类型的传播

构造模型时会将各种不同类型的模块连接起来，而这些不同类型的模块所支持的数据类型往往不完全相同。如果用直线连接在一起的两个模块所支持的数据类型（注意是指输出/输入信号的数据类型，而不是模块参数的数据类型）有冲突，那么当进行仿真、查看端口数据类型或是更新数据类型时就会弹出一个提示对话框，告诉用户出现冲突的信号和端口，而且有冲突的信号的路径会被加亮显示。这时就可以通过在有冲突的模块之间插入一个 DataTypeConversion 模块来解决类型冲突。

一个模块的输出一般是模块输入和模块参数的函数。而在实际建模过程中，输入信号的数据类型和模块参数的数据类型往往是不同的，Simulink 在计算这种输出时会把参数类型转换到信号的数据类型。但是如果出现下面的这些情形，就要进行不同的处理。

- 当信号的数据类型无法表示参数值时，Simulink 将中断仿真，并给出错误信息。

在如图 7-21 所示的示例模型，两个常数模块的输出信号类型设置为布尔型，而连续信号积分器只接受 double 类型信号，所以弹出出错提示框。这时可以在示例模型中插入（如果读者不熟悉该操作，参考第 7.2.1 节的内容）一个 DataTypeConversion 模块，并将其输出改成 double 数据类型就不会报错了，如图 7-22 所示。

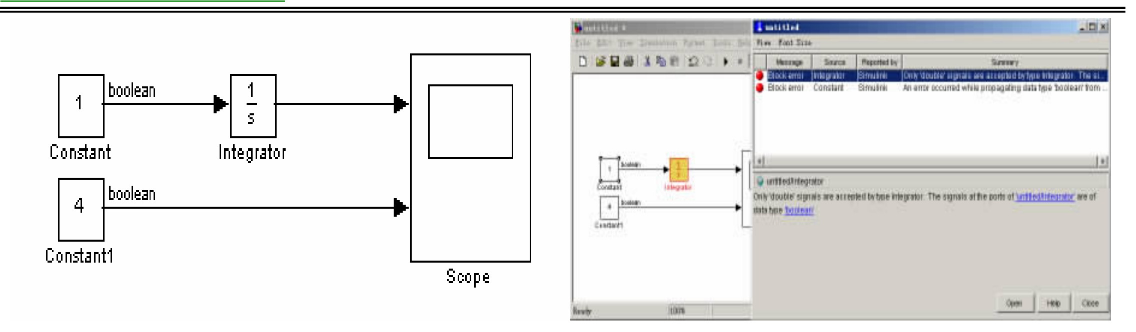


图 7-21　数据类型示例模型

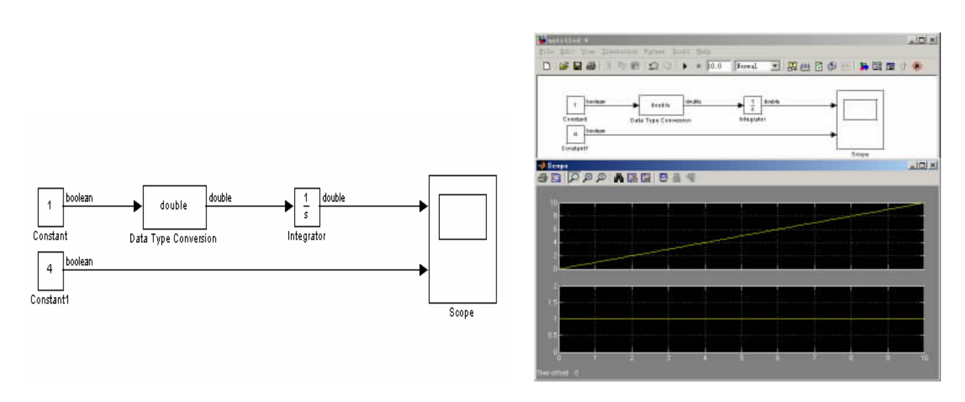


图 7-22　修改后的示例

- 如果信号的数据类型能够表示参数的值，仅仅是损失表示的精度，那么 Simulink 会继续仿真，并在 MATLAB 7.0 命令窗口中给出一个警告信息。

在如图 7-23 所示的图中，常数模块的常数值参数设置为 uint8（1），输出数据类型设为"Inherit from Constant value"；增益模块的增益参数设置为 double(3.2)，输出数据类型设为"Same as input"，参数数据类型设置为"Interit via internal rule"。Simulink 会把 3.2 的整数部分截取，并转换成无符号数，所以最后结果为 3。

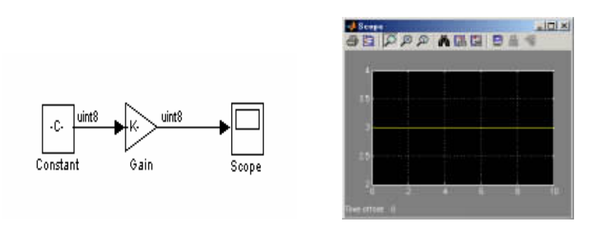


图 7-23　数据精度损失的示例

3．使用复数信号

Simulink 中默认的信号值都是实数，但在实际问题中有时需要处理复数信号。在 Simulink 中通常用下面示例中的两种方法来建立处理复数信号的模型，如图 7-24 所示。

- 向模型中加入一个 Constant 模块，将其参数设为复数；
- 分别生成复数的虚部和实部，再用 Real-Image to Complex 模块把它们联合成一个复数；
- 分别生成复数的幅值和幅角，再用 Magnitue-Angle to Complex 模块把它们联合成一个

复数。

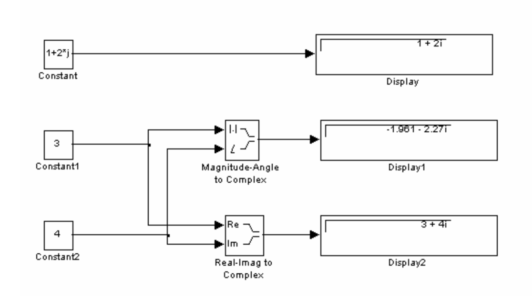


图 7-24　引入复数

同样可以在 Simulink 中利用相关的模块将一个复数分解成虚部和实部或是幅值和幅角，而且 Simulink 中有许多的模块都接受复数输入并对复数进行运算，请读者自己查阅帮助文件来了解这些模块。

> 注意：Real-Image to Complex 模块和 Magnitue-Angle to Complex 模块均在 Simulink 库的 Math Operations 子库中。

4．标量信号和向量信号

向量信号是多个信号的集合，对应着模块窗口中几条并行连线的合成。Simulink 中大多数模块的输出默认情况下为标量信号，当输入为向量信号而模块参数为标量时，Simulink 会自动进行匹配。接下来通过介绍两个示例来回答下面两个问题。

- 怎样让模块输出向量信号？
- 当一个模块的输入既有向量又有标量时，Simulink 如何计算输出？即标量扩展问题。

【示例 1】使模块输出向量信号

在如图 7-25 所示的左边第一个示例，第一个模型中 Constant 模块的常数值参数设为[1 3]，输出信号的宽度为 2，因此 Scope 有两条直线。第二个模型中 Sine Wave 模块的幅度设为[1 3]，频率设为[1 2]，输出一个两位的向量信号，因此 Scope 中有两条直线。

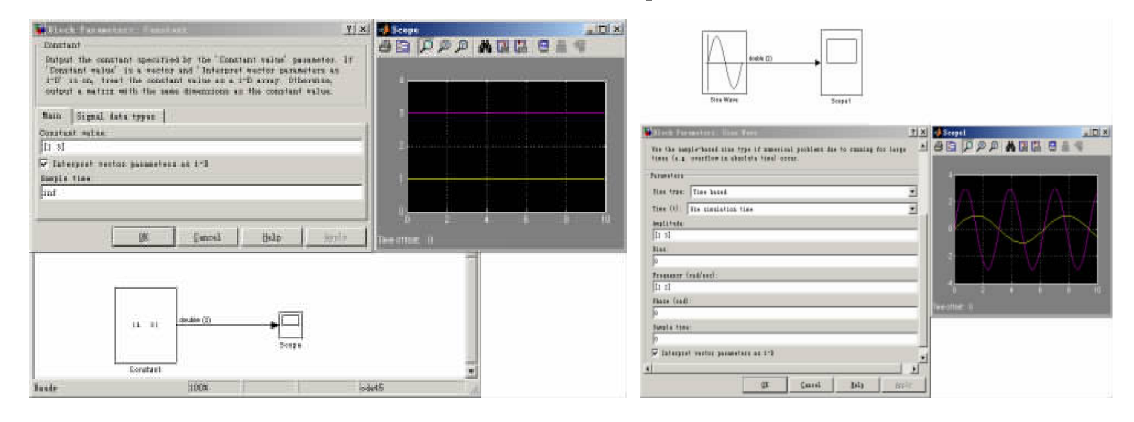


图 7-25　输出向量的模块示例

注意示例中 Sine Wave 的输出不是 4 位向量，模块中所有参数都是按向量元素的位置来

和信号对应，不存在自由组合的问题。读者在设置向量参数时一定要注意不同参数之间向量大小的匹配问题。上例中若将 Frequency 设为 1，不会出错；但若是设为[1 2 3]，就会报错。

【示例 2】既有向量又有标量输入的模块

在如图 7-26 所示的第二个示例中，由于输出信号的宽度设置为 2，因此 Scope 中显示有两条直线。Simulink 将 Constant 的 1 位标量信号扩展成 2 位的向量，新扩展位的值与原来的标量信号值相同。但是与上一例相似的是，如果一个信号是 3 位向量，一个是 2 位的向量，那么就会报错，因为 Simulink 无法确定扩展 2 位向量为 3 位向量时该以 2 位向量哪一位的值为标准。模块的参数也存在标量扩展的问题，读者只要多实践就会很容易掌握其中的规律。

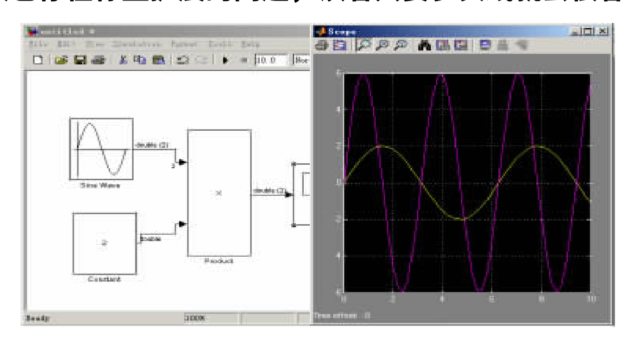


图 7-26　标量扩展的示例

7.1.6　Simulink 里的模块和模块库

用 Simulink 建模的过程，可以简单地理解为从模块库里选择合适的模块，然后将它们连接在一起，最后进行调试仿真。

模块库的作用就是提供各种基本模块，并将它们按应用领域以及功能进行分类管理以方便用户查找。库浏览器将各种模块库按树状结构进行罗列，以便用户快速地查询所需模块，同时它还提供了按名称查找的功能（如图 7-6 所示）。库浏览器中模块的多少取决于用户的安装，但至少应该有 Simulink 库，用户还可以自定义库。

模块是 Simulink 建模的基本元素，了解各个模块的作用是熟练掌握 Simulink 的基础。库中各个模块的功能可以在库浏览器中查到，如图 7-6 所示。下面详细介绍 Simulink 库中几个常用子库中的常用模块的功能，如表 7-8~表 7-17 所示，完整的介绍见本书附录。

表 7-8　　Commonly Used Blocks 子库

模块名	功能
Bus Creator	将输入信号合并成向量信号
Bus Selector	将输入向量分解成多个信号，输入只接受从 Mux 和 Bus Creator 输出的信号
Constant	输出常量信号
Data Type Conversion	数据类型的转换
Demux	将输入向量转换成标量或更小的标量
Discrete-Time Integrator	离散积分器模块
Gain	增益模块
In1	输入模块
Integrator	连续积分器模块
Logical Operator	逻辑运算模块

续表

模块名	功能
Mux	将输入的向量、标量或矩阵信号合成
Out1	输出模块
Product	乘法器，执行标量、向量或矩阵的乘法
Relational Operator	关系运算，输出布尔类型数据
Saturation	定义输入信号的最大和最小值
Scope	输出示波器
Subsystem	创建子系统
Sum	加法器
Switch	选择器，根据第二个输入信号来选择输出第一个还是第三个信号
Terminator	终止输出，用于防止模型最后的输出端没有接任何模块时报错
Unit Delay	单位时间延迟

表 7-9　　Continuous 子库

模块名	功能
Derivative	数值微分
Integrator	积分器与 Commonly Used Blocks 子库中的同名模块一样
State-Space	创建状态空间模型 dx/dt = Ax + Bu y = Cx + Du
Transport Delay	定义传输延迟，如果将延迟设置得比仿真步长大，可以得到更精确的结果
Transfer Fcn	用矩阵形式描述的传输函数
Variable Transport Delay	定义传输延迟，第一个输入接收输入；第二个输入接收延迟时间
Zero-Pole	用矩阵描述系统零点，用向量描述系统极点和增益

表 7-10　　Discontinuities 子库

模块名	功能
Coulomb& Viscous Friction	刻画在零点的不连续性，y = sign(x) * (Gain * abs(x) + Offset)
Dead Zone	产生死区，当输入在某一范围取值时输出为 0
Dead Zone Dynamic	产生死区，当输入在某一范围取值时输出为 0，与 Dead Zone 不同的是它的死区范围在仿真过程中是可变的
Hit　Crossing	检测输入是上升经过某一值还是下降经过这一值或是固定在某一值，用于过零检测
Quantizer	按相同的间隔离散输入
Rate Limiter	限制输入的上升和下降速率在某一范围
Rate Limiter Dynamic	限制输入的上升和下降速率在某一范围，与 Rate Limiter 不同的是它的范围在仿真过程中是可变的
Relay	判断输入与某两阈值的大小关系，当大于开启阈值，输出为 on；当小于关闭阈值时，输出为 off；当在两者之间时输出不变
Saturation	限制输入在最大和最小范围之内
Saturation Dynamic	限制输入在最大和最小范围之内，与 Saturation 不同的是它的范围在仿真过程之中是可变的
Wrap To Zero	当输入大于某一值时输出 0，否则输出等于输入

表 7-11　　Discrete 子库

模块名	功能
Difference	离散差分，输出当前值减去前一时刻的值
Discrete Derivative	离散偏微分
Discrete Filter	离散滤波器

续表

模块名	功能
Discrete State-Space	创建离散状态空间模型 x(n+1) = Ax(n) + Bu(n) y(n) = Cx(n) + Du(n)
Discrete Transfer Fcn	离散传输函数
Discrete Zero-Pole	离散零极点
Discrete-Time Integrator	离散积分器
First-Order Hold	一阶保持
Integer Delay	整数倍采样周期的延迟
Memory	存储单元，当前输出是前一时刻的输入
Transfer Fcn First Order	一阶传输函数，单位的直流增益
Zero-Order Hold	零阶保持

表 7-12　　Logic and Bit Operations 子库

模块名	功能
Bit Clear	将向量信号中某一位置为 0
Bit Set	将向量信号中某一位置为 1
Bitwise Operator	对输入信号进行自定义的逻辑运算
Combinatorial　Logic	组合逻辑，实现一个真值表
Compare To Constant	定义如何与常数进行比较
Compare To Zero	定义如何与零进行比较
Detect Change	检测输入的变化，如果输入的当前值与前一时刻的值不等，则输出 TRUE，否则为 FALSE
Detect Decrease	检测输入是否下降，是则输出 TRUE，否则输出 FALSE
Detect Fall Negative	若输入当前值是负数，前一时刻值为非负则输出 TRUE，否则为 FALSE
Detect Fall Nonpositive	若输入当前值是非正，前一时刻值为正数则输出 TRUE，否则为 FALSE
Detect Increase	检测输入是否上升，是则输出 TRUE，否则输出 FALSE
Detect Rise Nonnegative	若输入当前值是非负，前一时刻值为负数则输出 TRUE，否则为 FALSE
Detect Rise Positive	若输入当前值是正数，前一时刻值为非正则输出 TRUE，否则为 FALSE
Extract Bits	从输入中提取某几位输出
Interval Test	检测输入是否在某两个值之间，是则输出 TRUE，否则输出 FALSE
Logical Operator	逻辑运算
Relational Operator	关系运算
Shift Arithmetic	算术平移

表 7-13　　Math Operations 子库

模块名	功能
Abs	求绝对值
Add	加法运算
Algebraic Constraint	将输入约束为零，主要用于代数等式的建模
Assignment	选择输出输入的某些值
Bias	将输入加一个偏移，Y = U + Bias
Complex to Magnitude-Angle	将输入的复数转换成幅度和幅角
Complex to Real-Imag	将输入的复数转换成实部和虚部
Divide	实现除法或乘法
Dot Product	点乘
Gain	增益，实现点乘或普通乘法
Magnitude-Angle to Complex	将输入的幅度和幅角合成复数
Math Function	实现数学函数运算

续表

模块名	功能
Matrix Concatenation	实现矩阵的串联
MinMax	将输入的最小或最大值输出
Polynomial	多项式求值，多项式的系数以数组的形式定义
MinMax Running Resettable	将输入的最小或最大值输出，当有重置信号 R 输入时，输出被重置为初始值
Product of Elements	将所有输入实现连乘
Real-Imag to Complex	将输入的两个数当成一个复数的实部和虚部合成一个复数
Reshape	改变输入信号的维数
Rounding Function	将输入的整数部分输出
Sign	判断输入的符号，若为正输出 1，为负输出-1，为零输出 0
Sine Wave Function	产生一个正弦函数
Slider Gain	可变增益
Subtract	实现加法或减法
Sum	加法或减法
Sum of Elements	实现输入信号所有元素的和
Trigonometric Function	实现三角函数和双曲线函数
Unary Minus	一元的求负
Weighted Sample Time Math	根据采样时间实现输入的加法、减法、乘法和除法，只对离散信号适用

表 7-14　Ports & Subsystems 子库

模块名	功能
Configurable Subsystem	用于配置用户自建模型库，只在库文件中才可用
Atomic Subsystem	只包括输入/输出模块的子系统模板
CodeReuseSubsystem	只包括输入/输出模块的子系统模板
Enable	使能模块，只能用在子系统模块中
Enabled and Triggered Subsystem	包括使能和边沿触发模块的子系统模板
Enabled Subsystem	包括使能模块的子系统模板
For Iterator Subsystem	循环子系统模板
Function-Call Generator	实现循环运算模板
Function-Call Subsystem	包括输入/输出和函数调用触发模块的子系统模板
If	条件执行子系统模板，只在子系统模块中可用
If Action Subsystem	由 If 模块触发的子系统模板
Model	定义模型名字的模块
Subsystem	只包括输入/输出模块的子系统模板
Subsystem Examples	子系统演示模块，在模型中用鼠标左键双击该模块图标可以看到多个子系统示例
Switch Case	条件选择模块
Switch Case Action Subsystem	由 Switch Case 模块触发的子系统模板
Trigger	触发模块，只在子系统模块中可用
Triggered Subsystem	触发子系统模板
While Iterator Subsystem	条件循环子系统模板

表 7-15　Sinks 子库

模块名	功能
Display	显示输入数值的模块
Floating Scope	浮置示波器，由用户来设置所要显示的数据
Stop Simulation	当输入不为零时，停止仿真
To File	将输入和时间写入 MAT 文件

续表

模块名	功能
To Workspace	将输入和时间写入 MATLAB 7.0 工作空间中的数组或结构中
XY Graph	将输入分别当成 *X*、*Y* 轴数据绘制成二维图形

表 7-16 Sources 子库

模块名	功能
Band-Limited White Noise	有限带宽的白噪声
Chirp Signal	产生 Chirp 信号
Clock	输出当前仿真时间
Constant	输出常数
Counter Free-Running	自动计数器，发生溢出后又从 0 开始
Counter Limited	有限计数器，当计数到某一值后又从 0 开始
Digital Clock	以数字形式显示当前的仿真时间
From File	从 MAT 文件中读取数据
From Workspace	从 MATLAB7.0 工作空间读取数据
Pulse Generator	产生脉冲信号
Ramp	产生按某一斜率的数据
Random Number	产生随机数
Repeating Sequence	重复输出某一数据序列
Signal Builder	具有 GUI 界面的信号生成器，在模型中用鼠标左键双击模块图标可看到如图 7-27 所示的图形用户界面，利用该界面可以直观地构造各种信号
Signal Generator	信号产生器
Sine Wave	产生正弦信号
Step	产生阶跃信号
Uniform Random Number	按某一分布在某一范围生成随机数

表 7-17 User-Defined Functions 子库

模块名	功能
Fcn	简单的 MATLAB 7.0 函数表达式模块
Embedded MATLAB Function	内置 MATLAB 7.0 函数模块，在模型窗口用鼠标左键双击该模块图标就会弹出 M 文件编辑器
M-file S-Function	用户使用 MATLAB 语言编写的 S-函数模块
MATLAB Fcn	对输入进行简单的 MATLAB 函数运算
S-Function	用户按照 S-函数的规则自定义的模块，用户可以使用多种语言进行编写
S-Function Builder	具有 GUI 界面的 S-函数编辑器，在模型中用鼠标左键双击该模块图标可看到如图 7-28 所示的图形用户界面，利用该界面可以方便地编辑 S-函数模块
S-Function Examples	S-函数演示模块，在模型中用鼠标左键双击该模块图标可以看到多个 S-函数示例

除 Simulink 库之外，MATLAB 7.0 还为用户提供了大量不同应用领域的基本模块库。关于这部分内容，请读者通过前面介绍的方法在 Simulink 模块库浏览器中查看各个模块的功能。

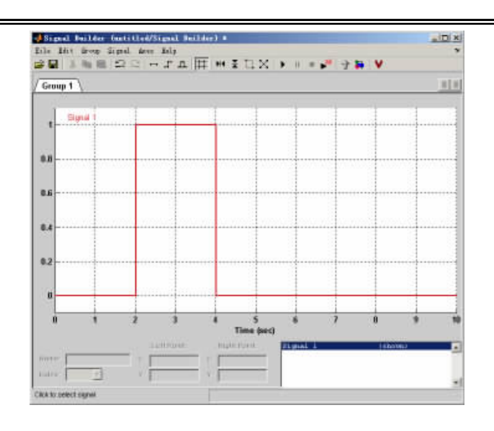


图 7-27 信号产生器图形用户界面

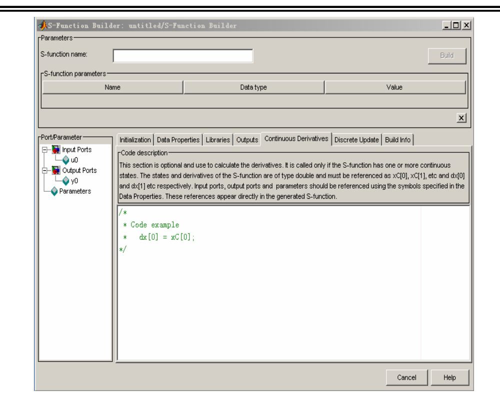


图 7-28 S-函数编辑器

7.2 模型的创建

7.2.1 Simulink 模块的基本操作

模块是建立 Simulink 模型的基本单元。利用 Simulink 进行系统建模就是用适当的方式把各种模块连接在一起。下面如表 7-18 和表 7-19 所示的这些表格汇总了 Simulink 里对模块、直线和信号标签进行各种操作的方法，其中有许多操作并不是惟一的，读者自己实践一下就可以摸索出其他的操作方法。

表 7-18 对模块进行操作

任务	Microsoft Windows 环境下的操作
选择一个模块	用鼠标左键单击要选择的模块，当用户选择了一个模块，那么之前选择的模块被放弃
选择多个模块	按住鼠标左键不放拖动鼠标，将要选择的模块包括在鼠标画出的方框里；或者按住 Shift，然后逐个选择
不同窗口间复制模块	直接将模块从一个窗口拖动到另一个窗口
同一模型窗口内复制模块	先选中模块，然后同时按下 Ctrl+C 键，再同时按下 Ctrl+V 键；还可以在选中模块后，通过快捷菜单来实现
移动模块	按下鼠标左键直接拖动模块
删除模块	先选中模块，再按下 Delete 键或者通过 Delete 菜单
连接模块	先选中源模块，然后按住 Ctrl 并用鼠标左键单击目标模块
断开模块间的连接	先按下 Shift 键，然后用鼠标左键拖动模块到另一个位置；或者也可以将鼠标指向连线的箭头处，当出现一个小圆圈圈住箭头时按下鼠标左键并移动连线
改变模块大小	先选中模块，然后鼠标移到模块方框的一角，当鼠标图标变成两端有箭头的线段时，按下鼠标左键拖动模块图标以改变图标大小
调整模块的方向	先选中模块，然后通过 Format→Rotate Block 菜单来改变模块方向
给模块加阴影	先选中模块，然后通过 Format→ Show Drop Shadow 菜单来改变模块方向
修改模块名	用鼠标左键双击模块名，然后修改
模块名的显示与否	先选中模块，然后通过 Format→ ShowName/Hide Name 菜单来决定是否显示模块名

续表

任务	Microsoft Windows 环境下的操作
改变模块名的位置	先选中模块，然后通过 Format→ Flip Name 菜单来改变模块名的显示位置
在连线之间插入模块	用鼠标拖动模块到连线上，使得模块的输入/输出端口对准连线

表 7-19　对直线进行操作

任务	Microsoft Windows 环境下的操作
选择多条直线	与选择多个模块的方法一样
选择一条直线	鼠标左键单击要选择的连线，当用户选择一条连线，那么之前选择的连线被放弃
连线的分支	按下 Ctrl 键，然后拖动直线；或者按下鼠标左键并拖动直线
移动直线段	按下鼠标左键直接拖动直线
移动直线顶点	将鼠标指向连线的箭头处，当出现一个小圆圈圈住箭头时按下鼠标左键移动连线
直线调整为斜线段	先单击“ Shift ”按钮，将鼠标指向需要移动的直线上的一点并按下鼠标左键直接拖动直线，如图 7-29 所示
直线调整为折线段	按住鼠标左键不放直接拖动直线

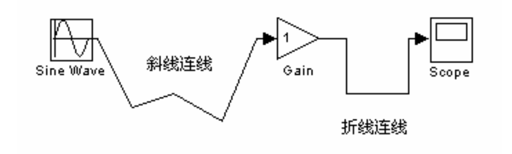


图 7-29　对连线的操作

Simulink 中大部分模块的参数（Parameters）都允许用户进行设置。用户只需双击要设置的模块或用鼠标右键单击模块并在弹出的上下文菜单中选择“ <block name> Parameters ”或先选中要设置的模块后再选择“ Edit→ <block name> Parameters ”就会弹出“ 参数设置 ”对话框，如图 7-30 所示。[Gain Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).] 是对模块功能的描述，每个模块的参数设置对话框都会有对自身功能描述的部分。图 7-30 中是一个增益模块，用户可以设置它的增益大小、采样时间和输出数据类型等参数。模块参数还可以通过 set_param 命令设置，这在后面的章节中再进行介绍。

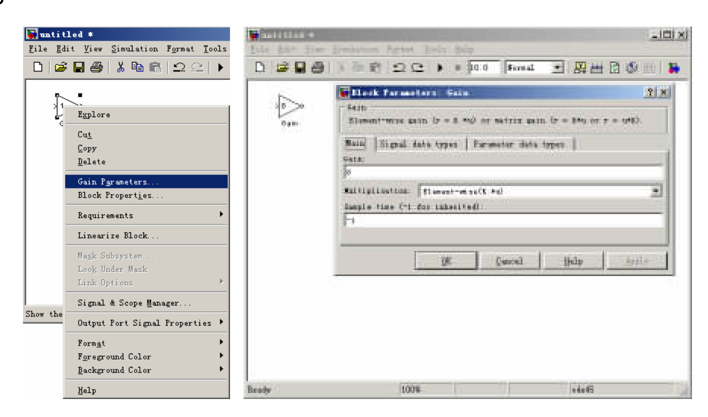


图 7-30　模块参数设置对话框

注意：“ <block name> Parameters ”中的<block name>是用户所选择的模块的名称。在图 7-30 的示例中，用户所选择的模块是 Gain，所以上下文菜单中要选择的是“ Gain Parameters ”。

Simulink 中的每个模块都有一个内容相同的属性（Properties）设置对话框，用鼠标右键单击模块并在弹出的上下文菜单中选择“Block Properties”或先选中要设置的模块后再选择“Edit→Block Properties”就会弹出属性设置对话框，如图 7-31 所示。“属性设置”对话框主要包括三项内容。

- “General”页

【说明（Description）】

用于对该模块在模型中的用法进行注释。

【优先级（Priority）】

规定该模块在模型中相对于其他模块执行的优先顺序。优先级的数值必须是整数，如果用户不输入数值，系统就会自动选取合适的优先级。优先级的数值越小（可以是负整数），优先级越高。一般都不需要设置它。

【标记（Tag）】

用户为模块添加的文本格式的标记。

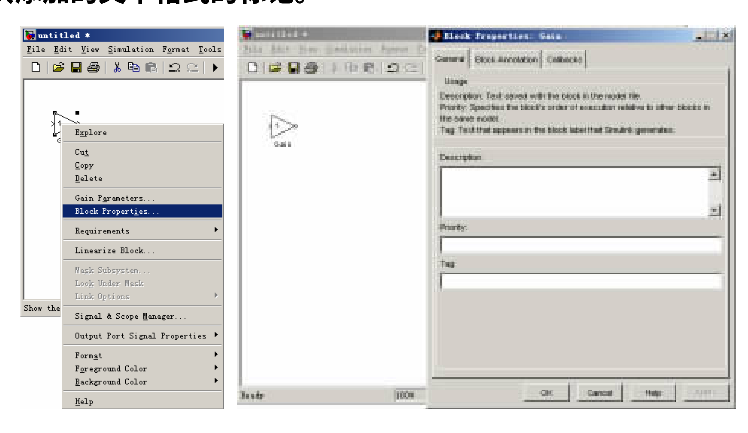


图 7-31　模块属性设置对话框

- “Block Annotation”页

用于指定在模块的图标下显示模块的哪个参数及其值，以什么格式显示。属性格式字符串由任意的文本字符串加嵌入式参数名组成。首先设置“General”页的 Priority 为 4，然后在“Block Annotation”页的“Enter text and tokens for annotation”文本框中输入“Priority=%<priority>\n Gain=%<gain>”，最后单击“Apply”，注意观察模块图标的变化，如图 7-32 所示。

如果参数的值不是字符串或数字，相应位置会显示 N/S（not supported）。如果参数名无效，该位置将显示“？？？”。

- “Callbacks”页

用于定义当该模块发生某种特殊行为时所要执行的 MATLAB 表达式，也称回调函数。关于回调函数在后面的章节再进行介绍。

对信号进行标注以及在模型图表上建立描述模型功能的注释文字，是一个好的建模习惯。信号标签和注释如图 7-33 所示。表 7-20 和表 7-21 列出了对标注和注释的具体操作方法。给一个信号添加标签，只需用鼠标左键在直线上双击，然后输入文字即可。建立模型注释与之类似，只要在模型窗口的空白处双击鼠标左键，然后输入注释文字即可。

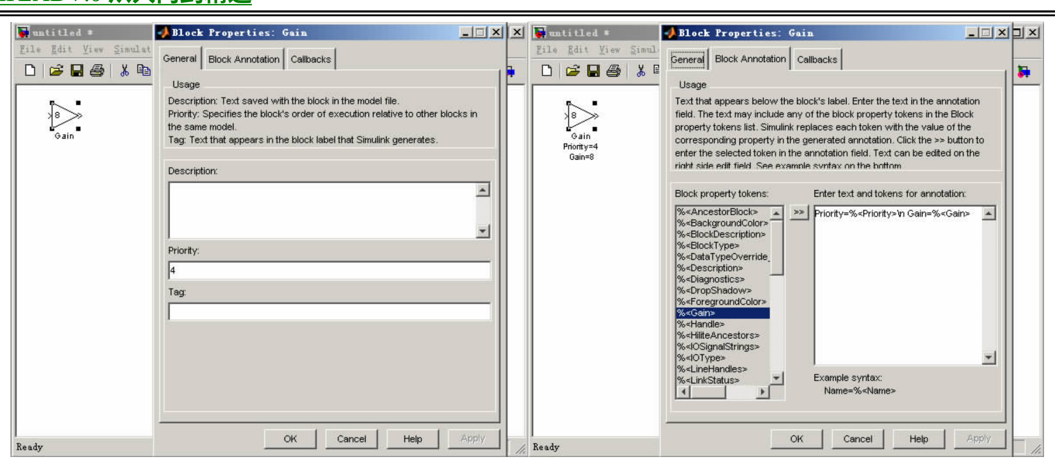


图 7-32　设置“ Block Annotation ”

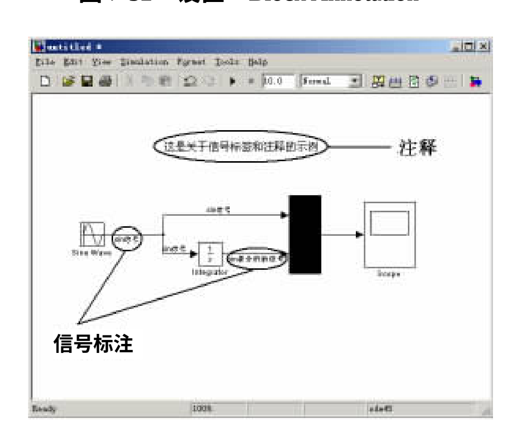


图 7-33　注释和信号标注示例

表 7-20　在连线上反映信息

任务	Microsoft Windows 环境下的操作
建立信号标签	在直线上直接用鼠标左键双击，然后输入
复制信号标签	按下 Ctrl 键，然后按住鼠标左键选中标签并拖动
移动信号标签	按住鼠标左键选中标签并拖动
编辑信号标签	在标签框内用鼠标左键双击，然后编辑
删除信号标签	按下 Shift 键，然后用鼠标左键单击选中标签，再按 Delete 键
用粗线表示向量	选择 Foamat→Port/Signal Displays→Wide Nonscalar Lines 菜单
显示数据类型	选择 Foamat→Port/Signal Displays→Port Data Types 菜单

表 7-21　对注释进行处理

任务	Microsoft Windows 环境下的操作
建立注释	在模型图标中用鼠标左键双击，然后输入文字
复制注释	按下 Ctrl 键，然后按下鼠标左键选中注释文字并拖动
移动注释	按下鼠标左键选中注释并拖动
编辑注释	单击注释文字，然后编辑
删除注释	按下 Shift 键，然后用鼠标选中注释文字，再按 Delete 键

和模块类似，每一个模型和每一条连线（代表信号）都有“参数设置”对话框和“属性设置”对话框，如图 7-34 所示（用鼠标右键单击模型的空白处选择相应的菜单就可以弹出对

话框)，它们的作用将在后面的章节进行介绍。

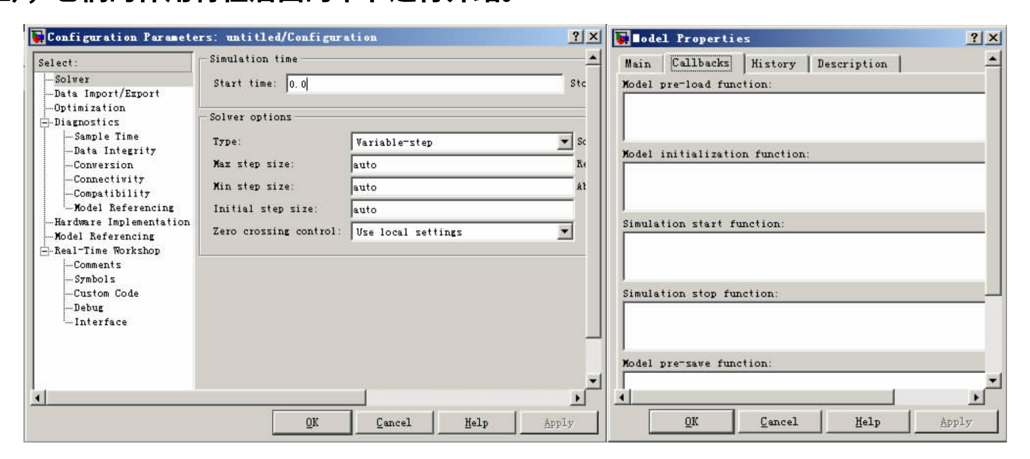


图 7-34　模型的仿真参数和“属性设置”对话框

7.2.2　创建模型的基本步骤

在本章的前面已向读者展示了一个简单的示例，通过该示例的学习，读者可能觉得使用 Simulink 建模实在是太简单了，只不过是用鼠标来选择几个模块，然后再把他们用几条线连接起来，最后按一下运行菜单观察结果曲线就可以了！但是当再次遇到实际问题时，读者可能会意识到所给的示例实在是一个太简单太过于理想化的模型，实际工程中要考虑的方面比这些要多得多，而且也要复杂得多，因此读者需要进一步学习和掌握 Simulink 中更为深层的内容。不过只要掌握了上一节的内容，读者就可以通过在线帮助来解决更为复杂的问题了。

在前面内容的基础上，下面向读者总结出使用 Simulink 进行系统建模和系统仿真的一般步骤：

- 画出系统草图。将所要仿真的系统根据功能划分成一个个小的子系统，然后用一个个小的模块来搭建每个子系统。这一步骤也体现了用 Simulink 进行系统建模的层次性特点。当然所选用的模块最好是 Simulink 的库里现有的模块，这样用户不必进行烦琐的代码编写，当然这要求用户必须熟悉这些库的内容；
- 启动 Simulink 模块库浏览器，新建一个空白模型；
- 在库中找到所需模块并拖到空白模型窗口中，按系统草图的布局摆放好各模块并连接各模块；
- 如果系统较复杂、模块太多，可以将实现同一功能的模块封装成一个子系统，使系统的模型看起来更简洁（封装的方法在第 7.3 节介绍）；
- 设置各模块的参数以及与仿真有关的各种参数（在后面的相关小节有详细介绍）；
- 保存模型，模型文件的后缀名为.mdl。
- 运行仿真，观察结果。如果仿真出错，请按弹出的错误提示框来查看出错的原因，然后进行修改；如果仿真结果与预想的结果不符，首先是要检查模块的连接是否有误、选择的模块是否合适，然后检查模块参数和仿真参数的设置是否合理。
- 调试模型。如果在上一步中没有检查出任何错误，那么就有必要进行调试（调试的方法在第 7.9 节中介绍），以查看系统在每一个仿真步的运行情况，直至找到出现仿真结

果与预想的或实际情况不符的地方，修改后再进行仿真，直至结果符合要求。当然最后还要保存模型。

7.2.3 模型文件格式

前面的示例模型都是通过图形界面来建立的，Simulink 还为用户提供了通过命令行建立模型和设置模型参数的方法。一般情况下，用户不需要使用这种方式来建模，因为它要求用户熟悉大量的命令，而且很不直观。这一节只对 Simulink 的模型文件格式作一个粗略的介绍，用户若在实际建模时遇到相关的问题可以查阅在线帮助。

Simulink 将每一个模型（包括库）都保存在一个以.mdl 为后缀的文件里，称为模型文件。一个模型文件就是一个结构化的 ASCII 文件，它包括关键字和各种参数的值。下面以一个示例来介绍如何查看模型文件和模型文件的结构。

- 建立如图 7-35 所示的模型，并保存为 modelfile.mdl；
- 在 MATLAB 7.0 窗口中通过"File→New→M-File"菜单打开 MATLAB Edit；
- 在 MATLAB Edit 窗口中通过"File→Open"菜单打开刚刚保存的 modelfile.mdl 文件，如图 7-35 所示。如果在目录中没有看到该文件，请查看文件打开对话框的文件类型是否为所有文件。

由示例中的模型文件可以看出，模型文件按照模块继承的层次关系描述模型的各个组件。文件的结构如下：

```
Model {
  <Model Parameter Name> <Model Parameter Value>
  ...
  BlockDefaults {
    <Block Parameter Name> <Block Parameter Value>
    ...
  }
  AnnotationDefaults {
    <Annotation Parameter Name> <Annotation Parameter Value>
    ...
  }
  System {
    <System Parameter Name> <System Parameter Value>
    ...
    Block {
      <Block Parameter Name> <Block Parameter Value>
      ...
    }
    Line {
      <Line Parameter Name> <Line Parameter Value>
      ...
```

```
            Branch {
              <Branch Parameter Name> <Branch Parameter Value>
              ...
            }
          }
          Annotation {
            <Annotation Parameter Name> <Annotation Parameter Value>
            ...
          }
        }
      }
```

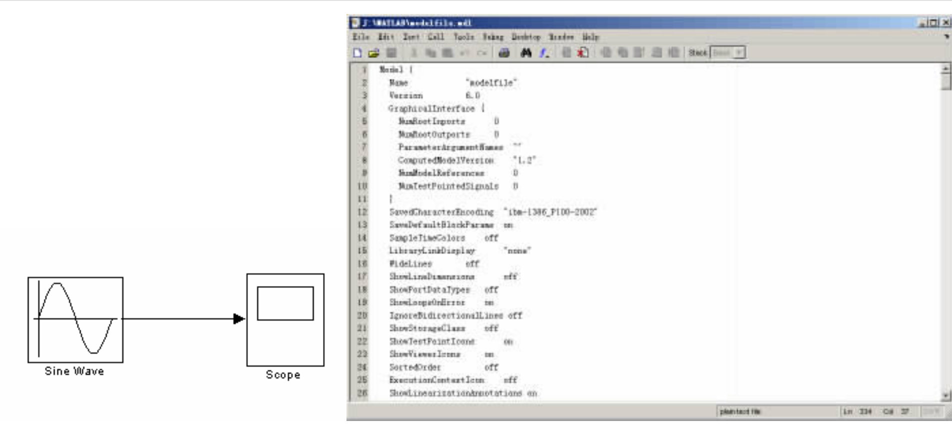


图 7-35　模型文件示例

文件分成下面几个部分来描述模型。

- Model 部分：用来描述模型参数，包括模型名称、模型版本和仿真参数等；
- BlockDefaults 部分：用来描述模块参数的默认设置；
- AnnotationDefaults 部分：用来描述模型的注释参数的默认值，这些参数值不能用 set_param 命令来修改；
- System 部分：用来描述模型中每一个系统（包括顶层的系统和各级子系统）的参数。每一个 System 部分都包括模块、连线和注释等。

7.3　子系统及其封装

7.3.1　创建子系统

1．为何创建子系统

当模型变得越来越大、越来越复杂时，用户很难轻易读懂以前所建的模型。在这种情况

下，通过子系统可以把大的模型分割成几个小的模型系统以使整个模型更简洁、可读性更高，而且这种操作也不复杂。建立子系统有以下优点：

- 减少模型窗口中模块的个数，使得模型窗口更加整洁；
- 把一些功能相关的模块集成在一起，还可以实现复用；
- 通过子系统可以实现模型图表的层次化，这样用户既可以采用自上而下的设计方法，也可以采用自下而上的设计方法。

2．如何创建子系统

在 Simulink 中有两种创建子系统的方法。

- 通过子系统模块来创建子系统：先向模型中添加 Subsystem 模块，然后打开该模块并向其中添加模块。
- 组合已存在的模块创建子系统：具体建立步骤见下一节的示例。

3．创建子系统示例

下面通过两个示例来介绍上文中提到的两种建立子系统的方法。

【示例 1】通过 Subsystem 模块来创建子系统

具体步骤如下：

- 从 Ports&Subsystems 中复制 Subsystem 模块到自己的模型中，如图 7-36 所示；
- 用鼠标左键双击 Subsystem 模块图标打开如图 7-36 所示 Subsystem 模块编辑窗口；
- 在新的空白窗口创建子系统，然后保存；
- 运行仿真并保存。

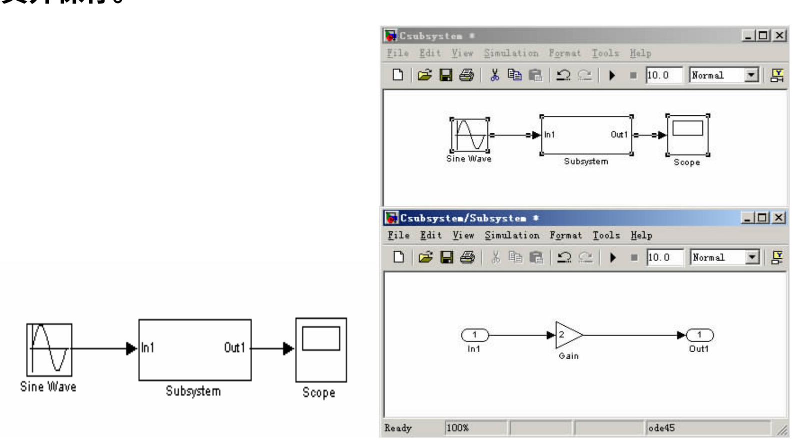


图 7-36　通过 Subsystem 模块创建子系统

【示例 2】组合已存在的模块创建子系统

具体步骤如下：

- 创建如图 7-37 所示的系统；
- 选中要创建成子系统的模块，如图 7-37 所示；
- 选择【Edit-> Create Subsystem】菜单，结果如图 7-38 所示；
- 运行仿真并保存。



图 7-37　组合已存在的模块创建子系统　　　　图 7-38　创建子系统示例

7.3.2　封装子系统

子系统可以使模型更简洁，但是在设置模型中各模块的参数时仍然很繁琐，此时可以使用封装技术。封装可以使用户创建的子系统表现得与 Simulink 提供的模块一样拥有自己的图标，并且用鼠标左键双击模块图标时会出现一个用户自定义的“参数设置”对话框，实现在一个对话框中设置子系统中所有模块的参数。

1．为何封装

封装子系统可以为用户带来很多好处。

- 在设置子系统中各个模块的参数时只通过一个参数对话框就可以完成所需设置；
- 为子系统创建一个可以反映子系统功能的图标；
- 可以避免用户在无意中修改子系统中模块的参数。

2．如何封装

用户可以按照如下的步骤来封装一个子系统。

- 选择需要封装的子系统；
- 选择“Edit→Edit mask”菜单，这时会弹出如图 7-39 所示的封装编辑器，通过它进行各种设置。编辑器中各项参数的含义在下一节的示例中进行介绍；
- 单击“Apply”或“OK”按钮保存设置。

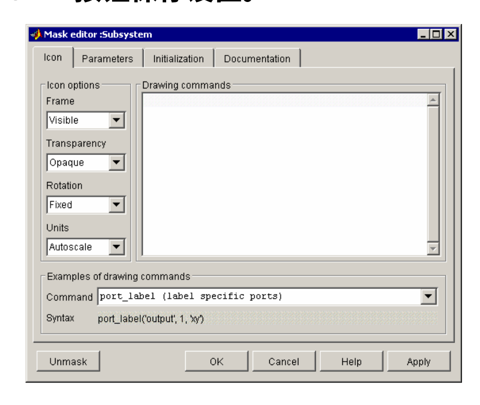


图 7-39　封装编辑器

3．封装示例

封装的操作很简单，也比较容易理解，下面通过一个简单的示例来介绍如何进行封装。

- 建立如图 7-40 所示的含有一子系统的模型，并设置子系统中 Gain 模块的 Gain 参数为一变量 m；
- 选中模型中的 Subsystem 子系统，选择“Edit→Edit mask”菜单（或用鼠标右键单击子系统弹出上下文菜单，选择“Edit mask”菜单）打开封装编辑器，如图 7-41 所示；
- 进行封装设置。按照如图 7-41 所示设置【Icon 页】；

【Icon 页】允许用户定义封装子系统的图标，其中各项设置的含义如下：

【Icon options 面板】：定义图标的边框是否可见（Frame），系统在图标中自动生成的端口标签等是否可见（Transparency.）等，用户只要试一试就会很快理解掌握。

【Drawing commands 文本框】：用 MATLAB 命令来定义如何绘制模型的图标，这里的绘图命令可以调用【Initialization 页】中定义的变量。

【Examples of drawing commands 面板】：向用户解释如何使用各种绘制图标的命令，每种命令都对应在右下角有一个示例。用户可以方便地按照“command”选项框中的命令格式和右下角给出的相应示例图标来书写自己的图标绘制命令。

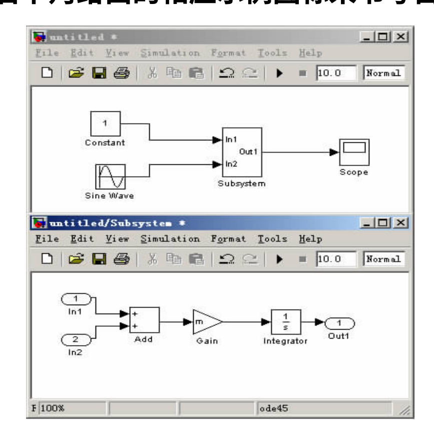


图 7-40　封装子系统示例

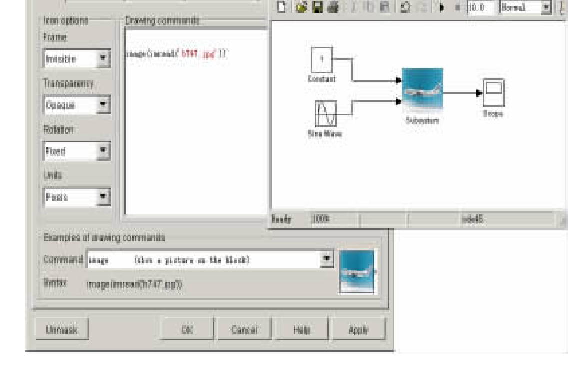


图 7-41　设置图标

- 按照如图 7-42 所示设置“Parameters”页

“Parameters”页允许用户定义封装子系统的参数对话框的可设置参数，其中各项设置的含义如图 7-43 所示。

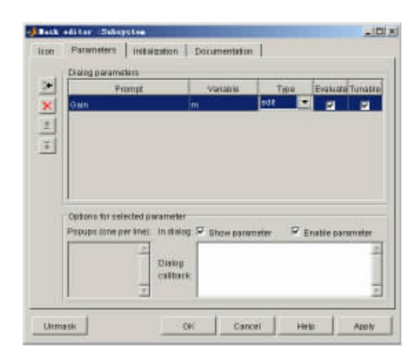


图 7-42　设置参数

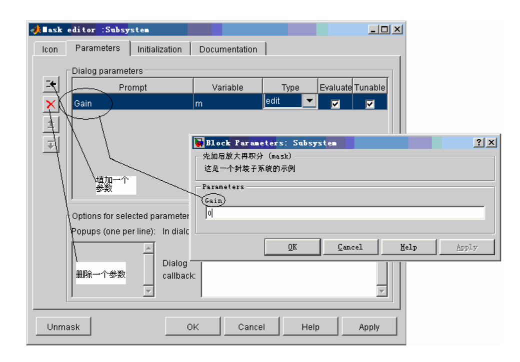


图 7-43　参数设置的含义

- 按照如图 7-44 所示设置“Initialization”页

“Initialization”页允许用户定义封装子系统的初始化命令。初始化命令可以使用任何有

效的 MATLAB 表达式、函数、运算符和在“Parameters”页定义的变量，但是初始化命令不能访问 MATLAB 7.0 工作空间的变量。在每一条命令后用分号结束可以避免模型运行时在 MATLAB 7.0 命令窗口显示运行结果。一般在此定义附加变量、初始化变量或绘制图标等。

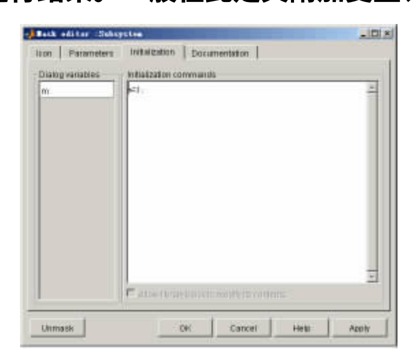
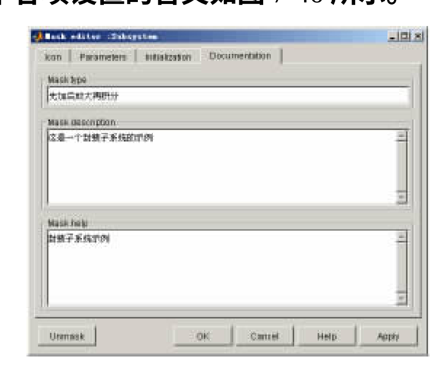


图 7-44　设置初始化参数

- 按照如图 7-45 所示设置“Documentation”页

“Documentation”页允许用户定义封装子系统的封装类型、模块描述和模块帮助信息，其中各项设置的含义如图 7-46 所示。



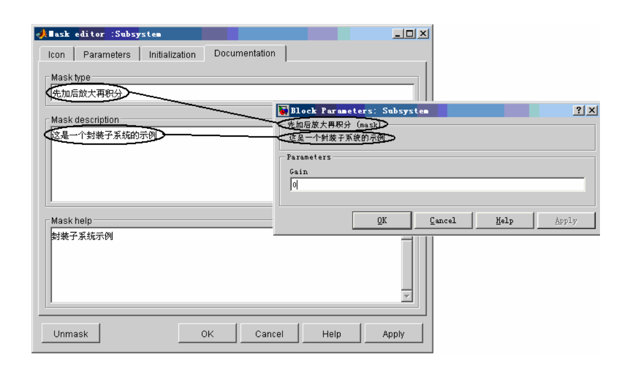


图 7-45　设置“Documentation”页参数　　图 7-46　“Documentation”设置的含义

单击“Apply”或“OK”按钮。用鼠标左键双击模型中的 Subsystem 子系统，弹出如图 7-47 所示封装子系统参数设置对话框。封装子系统的模块对话框中的变量不可在 MATLAB 工作空间赋值，这与非封装子系统不同。封装子系统有一个独立于 MATLAB 工作空间的内部存储空间。



图 7-47　封装后的子系统的参数对话框

- 运行仿真，用鼠标左键双击模型中的 Scope 模块，看到如图 7-48 所示的结果。读者也可以试着改变图 7-47 中 Gain 参数的值，然后观察输出是否有所改变，并考虑其原因。

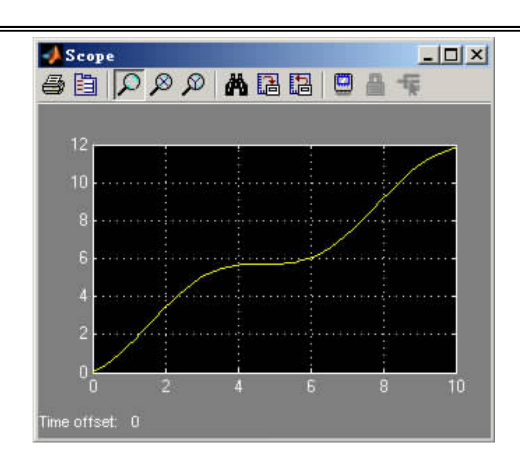


图 7-48　示例模型仿真结果

> **注意：如果在保存上面示例的模型时弹出 error 提示框，有可能是用户的 MATLAB 7.0 或 simulink 在封装技术上不支持中文，这时需要将“Documentation”页中的中文说明改成英文。**

7.3.3　定义自己的模块库

1．为何创建自己的模块库

当用户创建了很多封装子系统模块时，用户有必要分门别类地来存储这些模块。另外在进行仿真建模时，为了减少到各种子库中来回查寻所需模块，用户有必要将同一类功能的一组常用模块统一放置在同一模块库中。

2．如何创建自己的模块库

通过选择 Simulink 用户界面的“File→New→Library”菜单命令来创建模块库。选中该命令后弹出一个空白的库窗口，然后将需要存放在同一模块库中的模块复制到模块库窗口中即可，如图 7-49 所示。创建好模型后，用户在创建模型时可以不需要打开 Simulink 模块库浏览器，而只需要在 MATLAB 7.0 命令窗口输入存放相应模块的模块库的文件名即可。

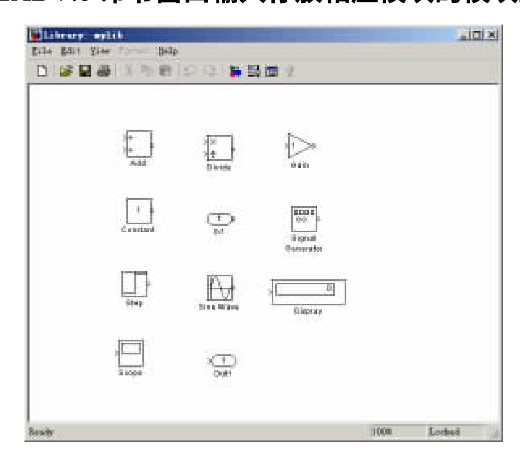


图 7-49　自建模型库

7.4　过零检测

当仿真一个动态系统时，Simulink 在每一个时间步使用过零检测技术来检测系统状态变量的间断点。如果 Simulink 在当前的时间步内检测到了不连续的点，那么它将找到发生不连续的精确时间点，并且会在该时间点的前后增加附加的时间步。本节将向读者介绍过零检测的重要性以及它的工作原理。

1．过零检测的重要性

在动态系统的运行过程中，状态的不连续经常会发生重要的事件，如当一个小球与地面发生碰撞时，它的位置就会产生急剧的变化。不连续常常会导致动态系统的显著变化，因此对不连续点进行精确的仿真非常重要，否则会导致仿真得到错误的系统行为。例如仿真碰撞中的小球，如果小球与地面碰撞的时间点发生在仿真时间步内，模型中的小球会在半空中改变方向，这会使研究者得到与物理学上的规律相违背的结论。

为了避免得到错误的结论，使不连续点发生的时刻成为仿真的一个时间点很重要。对于一个纯粹靠求解器来决定仿真时间的仿真器很难有效地做到上面的要求。

固定步长的求解器在整数倍时间步长的时间点上计算状态变量的值，然而这并不能保证有仿真时间点发生在状态不连续的时间点。当然用户可以通过减小时间步长使状态不连续的时间点成为一个仿真时间点，但是这会减慢仿真速度。

可变步长的求解器可以保证有仿真时间点发生在状态不连续的时间点。它可以动态地改变步长，当状态变量变化慢时增加步长，变化快时减小步长。在不连续点附近，系统状态发生剧烈变化，因此理论上这种求解器可以精确地找到不连续发生的时间点，但是这会增加仿真的时间点从而使仿真速度变慢。

2．过零检测的工作原理

使用过零检测技术，一个模块能够通过 Simulink 注册一系列过零变量，每一个变量就是一个状态变量（含有不连续点）的函数。当相应的不连续发生时，过零函数从正值或负值传递零值。在每一个仿真步结束时，Simulink 通过调用每一个注册了过零变量的模块来更新变量。然后 Simulink 检测是否有变量的符号发生改变（相对于上一仿真时间点的结果），如果有改变就说明当前时间步有不连续发生。

如果检测到过零点，Simulink 就会在每一个发生符号改变的变量的前一时刻值和当前值之间内插入新值以评估过零点的个数，然后逐步增加内插点数目并使其值依次越过每一个过零点。通过该技术，Simulink 避免精确地在不连续发生点进行仿真（不连续点处的状态变量的值可能没有定义）。

过零检测使得 Simulink 可以精确地仿真不连续点而不必通过减小仿真步长增加仿真点来实现，因此仿真速度不会受太大影响。大多数 Simulink 模块都支持过零检测，表 7-22 列出了 Simulink 中支持过零检测的模块。如果用户需要显式定义过零事件，可以使用 Discontinuities 子库中的 Hit Crossing 模块来实现。

表 7-22　　持过零点检测的模块

模块名	说明
Abs	一个过零检测：检测输入信号沿上升或下降方向通过零点
Backlash	两个过零检测：一个检测是否超过上限阈值，一个检测是否超过下限阈值
Dead Zone	两个过零检测：一个检测何时进入死区，一个检测何时离开死区
Hit Crossing	一个过零检测：检测输入何时通过阈值
Integrator	若提供了 Reset 端口，就检测何时发生 Reset；若输出有限，则有三个过零检测即检测何时达到上限饱和值、检测何时达到下限饱和值和检测何时离开饱和区
MinMax	一个过零检测：对于输出向量的每一个元素，检测一个输入何时成为最大或最小值
Relay	一个过零检测：若 relay 是 off 状态，就检测开启点；若是 on 状态，就检测关闭点
Relational Operator	一个过零检测：检测输出何时发生改变
Saturation	两个过零检测：一个检测何时达到或离开上限，一个检测何时达到或离开下限
Sign	一个过零检测：检测输入何时通过零点
Step	一个过零检测：检测阶跃发生时间
Switch	一个过零检测：检测开关条件何时满足
Subsystem	用于有条件地运行子系统：一个使能端口，一个触发端口

如果仿真的误差容忍度设置得太大，那么 Simulink 有可能检测不到过零点。例如，如果在一个时间步内存在过零点，但是在时间步的开始和最终时刻没有检测到符号的改变（如图 7-50 所示），那么求解器将检测不到过零点。

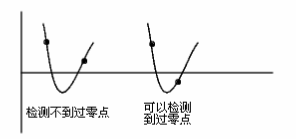


图 7-50　过零点检测

7.5　代数环

1．何为代数环及代数环如何产生

有些 Simulink 模块的输入端口是支持直接馈入的，这意味着这些模块的输出信号值在不知道输入端口的信号值之前就不能被计算出来。当一个支持直接馈入的输入端口由同一模块的输出直接或间接地通过由其他模块组成的反馈回路的输出驱动时，就会产生一个代数环，如图 7-51 所示。

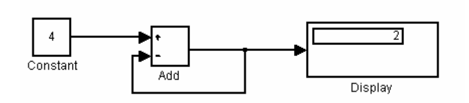


图 7-51　代数环

从代数的角度来看，图 7-51 中的模块的解是 $z = 2$，但是大多数的代数环是无法直接看出解的。Algebraic Constraint 模块为代数方程等式建模及定义其初始猜想值提供了方便，它

约束输入信号 F(*z*)等于零并输出代数状态 Z，其输出必须能够通过反馈回路影响输入。用户可以为代数环状态提供一个初始猜想值以提高求解代数环的效率。

一个标量代数环代表一个标量等式或是一个形如 F(*z*) = 0 的约束条件，其中 *z* 是环中一个模块的输出，函数 F 由环路中的另一个反馈回路组成。可将图 7-51 所示的含有反馈环的模型改写成用 Algebraic Constraint 模块创建的模型，其仿真结果不变，如图 7-52 所示。

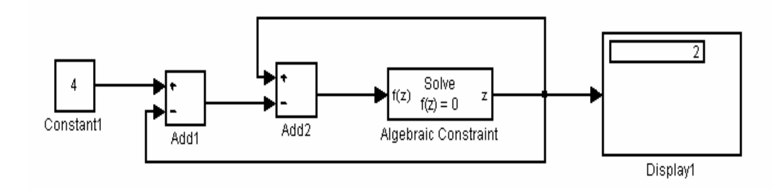


图 7-52 Algebraic Constraint 模块创建代数环模块

创建向量代数环也很容易，在如图 7-53 所示的向量代数环可用下面的代数方程描述：

$z2 + z1 - 1 = 0$

$z2 - z1 - 1 = 0$

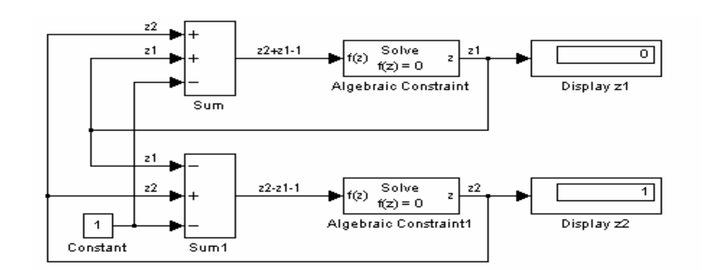


图 7-53 向量代数环

当一个模型包含一个 Algebraic Constraint 模块时就会产生一个代数环，这种约束可能是系统的物理连接的结果，或是用户试图为一个微分/代数系统（DAE）建模的结果。

2．如何求解代数环及如何消除代数环

为了求解 F(*z*) = 0，Simulink 环路求解器会采用弱线性搜索的秩为 1 的牛顿方法更新偏微分 Jacobian 矩阵。尽管这种方法很有效，但是如果代数状态 Z 没有一个好的初始估计值，解法器有可能不收敛。此时用户可以为代数环中的某个连线（对应一个信号）定义一个初始值，设置的办法有两个：可以通过 Algebraic Constraint 模块的参数设置；还可以通过在连线上放置 IC 模块（初始信号设置模块）。

当一个系统包含有代数环时，Simulink 会在每一个时间步进行循环求解。如果有可能，环路求解器会采用迭代的办法来求解，因此仿真速度会比较慢。

当一个模型中含有 Atomic Subsystem、Enabled Subsystem 或 Model 模块时，Simulink 可以通过模块的参数设置来消除其中一些代数环。对于含 Atomic Subsystem 和 Enabled Subsystem 模块的模型，可在模块参数设置对话框中选择‘ Minimize algebraic loop occurrences ”项；对于含 Model 模块的模型，可在“ configuration parameters 对话框 ”的“ Model Referencing 面板 ” 中选择 “ inimize algebraic loop occurrences ” 项。

7.6 回调函数

在前面介绍“模块属性设置”对话框时提到过回调函数（如图 7-32 所示的“设置模块属性”对话框）。所谓回调函数，是指当用户定义的模型或模块的图标发生某种特殊行为时所要执行的 MATLAB 表达式（M 文件和 M 文件函数）。

为模型或模块设置回调函数的方法有下面两种：

- 通过模型或模块的属性对话框来设置；
- 通过 MATLAB 相关的命令来设置。

在如图 7-54 所示的“模块属性设置”对话框中的 Callbacks 页列出了可以为模块设置回调函数的参数（每一个参数对应模块的一种行为），表 7-23 列出了这些参数的含义。可以为模型设置回调函数的参数与可设置的模块参数类似，如表 7-24 所示。图 7-55 是“模型属性设置”对话框。

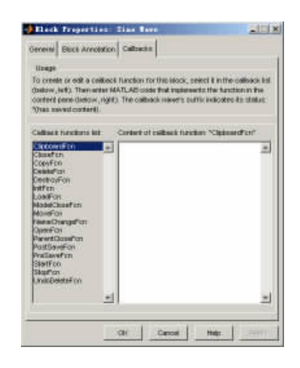


图 7-54 为模块设置函数的参数

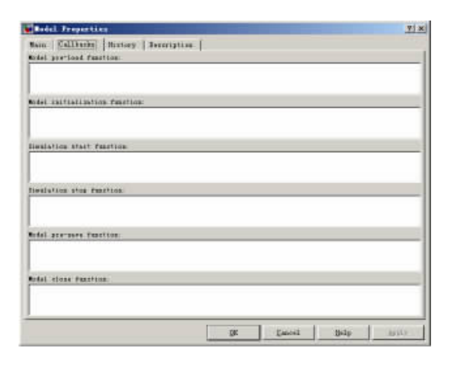


图 7-55 模型属性对话框

表 7-23 模块的回调参数

模块回调参数名称	参数含义
ClipboardFcn	在模块被复制或是剪切到系统粘贴板时
CloseFcn	当使用 close_system 命令关闭模块时
CopyFcn	模块被复制之后，该回调对于子系统是递归的。如果是使用 add_block 命令复制模块，该回调也会被执行
DeleteFcn	在模块被删除之前
DeleteChildFcn	在模块从子系统中被删除之后
DestroyFcn	当模块已经被毁坏时
InitFcn	在模块被编译和模块参数被估值之前
LoadFcn	在模块被载入之后，该回调对于子系统是递归的
ModelCloseFcn	在模块被关闭之前，该回调对于子系统是递归的
MoveFcn	在模块被移动或调整大小时
NameChangeFcn	在模块的名字或路径发生改变时
OpenFcn	在模块被打开时，一般用于子系统模块。当用户双击打开模块或者使用 open_system 命令打开模块时调用
ParentCloseFcn	在关闭包含该模块的子系统或者是用 new_system 命令建立的包含该模块的子系统时
PostSaveFcn	在模块被保存之后，该回调对于子系统是递归的
PreSaveFcn	在模块被保存之前，该回调对于子系统是递归的
StartFcn	在模块被编译之后，仿真开始之前

续表

模块回调参数名称	参数含义
StopFcn	在仿真结束时
UndoDeleteFcn	当一个模块的删除操作被取消时

表 7-24　　　　　　　　　　　模型的回调参数

模型回调参数名称	参数含义
CloseFcn	在模型图表被关之前调用
PostLoadFcn	在模型被载入之后调用
InitFcn	在模型的仿真开始时调用
PostSaveFcn	在模型被保存之后调用
PreLoadFcn	在模型被载入之前调用，用于预先载入模型使用的变量
PreSaveFcn	在模型被保存之前调用
StartFcn	在模型仿真开始之前
StopFcn	在模型仿真停止之后，在 StopFcn 执行前，仿真结果先被写入工作空间中的变量和文件中

下面通过一个示例介绍为模型或模块设置回调函数的两种方法的具体实现过程。

【第一种方法】

- 首先建立一个如图 7-56 所示的模型 example；
- 设置模型属性对话框“CallBack”页的 Simulation stop function 文本为 plot(tout,yout)；
- 运行仿真，结束时会自动弹出图形窗口。

同样也可以通过设置 Out1 模块的 Callback 属性来达到这一目的，如图 7-57 所示。

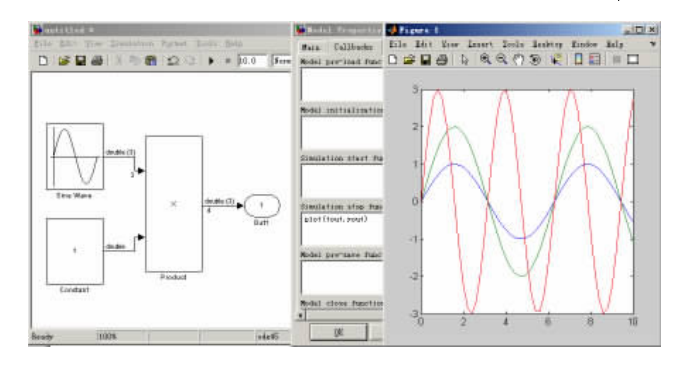


图 7-56　通过模型属性对话框为模型设置回调函数

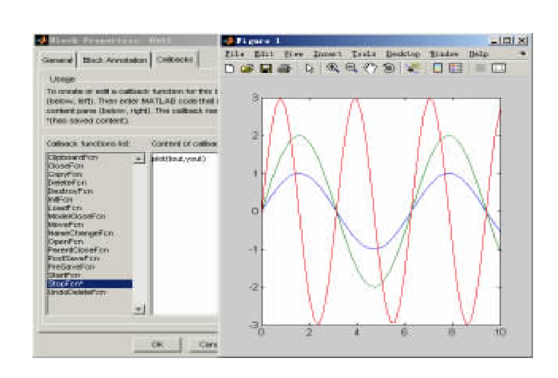


图 7-57　通过模块属性对话框为模型设置回调函数

【第二种方法】

- 首先建立好如图 7-56 所示的模型；
- 设置 Constant 模块参数为变量 con，Sine Wave 的幅度和频率参数设置为 3 位向量；
- 在 MATLAB 7.0 命令窗口输入下面的代码：

```
set_param('example','PreLoadFcn','ini_con');
set_param('example','StopFcn','out_plot');
```

- 保存模型之后，新建一个名为 ini_con 的 M 文件，其内容为 con = 3；
- 再新建一个名为 out_plot 的 M 文件，其内容为 plot(tout, yout)；
- 运行仿真，结果如图 7-58 所示。

> 注意：如果读者在运行示例时出错，请检查你的当前目录是不是你保存模型的目录。如果不是，请将当前目录改成保存模型的目录，再运行上面两条 set_param 命令，最后运行仿真。

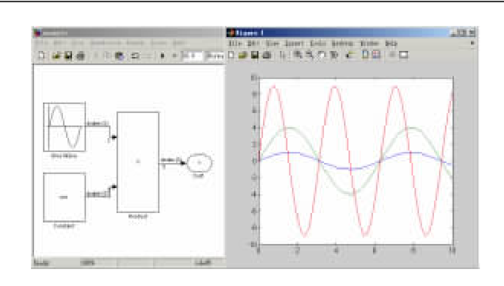


图 7-58　通过 MATLAB 命令为模型设置回调函数

7.7　运行仿真

建好一个模型之后就要运行模型和分析仿真结果，前面也简单地提到过运行仿真，本节将对启动仿真的方法及各种仿真参数的设置进行详尽地讲解。

7.7.1　使用窗口运行仿真

当建立好模型后，可以直接在模型窗口通过菜单或工具栏进行仿真，如图 7-59 所示。可以通过这些菜单设置仿真的参数、仿真的时间以及选择解法器等，而不需要去记忆 MATLAB 的各种命令语法。前面第 7.1.2 节已对模型窗口的菜单和工具栏作了详尽介绍，读者加以实践即可掌握。

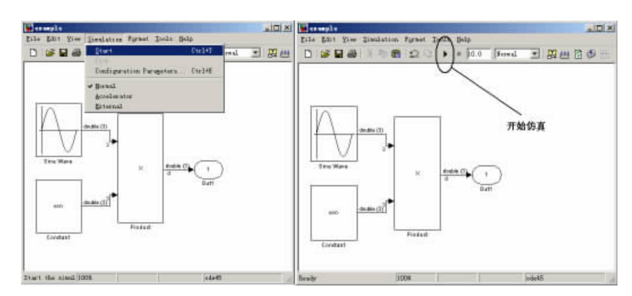


图 7-59　通过菜单命令或工具栏按钮运行仿真

7.7.2　仿真参数的设置

1．设置仿真参数和选择解法器

可以通过模型窗口的“Simulation→ Configuration Parameters”命令打开设置仿真参数的对话框，也可以通过上下文菜单的“Configuration Parameters”项（单击模型窗口中的空白处也可以打开）打开，如图 7-60 所示。

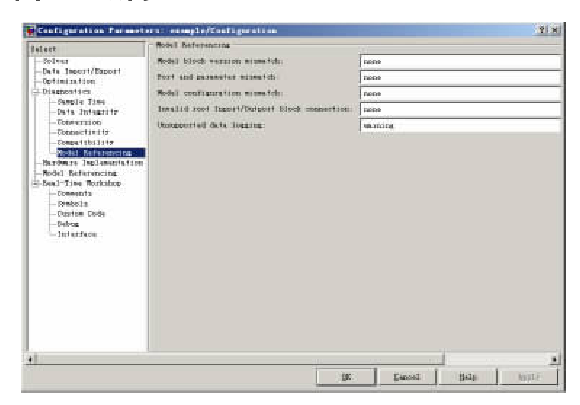


图 7-60　仿真参数对话框

对话框将参数分成不同类型的 6 组，下面对每一组中各个参数的作用和设置方法进行简单的介绍。

（1）【Solver 面板】

该面板主要用于设置仿真开始和结束时间，选择解法器，并设置它的相关参数，如图 7-61 所示。

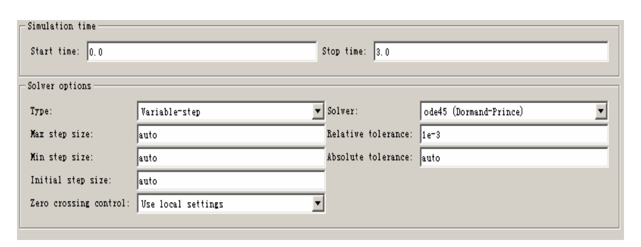


图 7-61　Solver 面板

Simulink 支持两类解法器：固定步长和可变步长解法器。两种解法器计算下一个仿真时间的方法都是在当前仿真时间上加一个时间步长。不同的是，固定步长解法器的时间步长是常数，而可变步长解法器的时间步长是根据模型动态特性可变的。当模型的状态变化特别快时，为了保证精度要降低时间步长，反之就增加时间步长。面板中 Type 用于设置解法器的类型，当选择了不同的类型时，Solver 中可选的解法器列表也不同，有关各种求解器的使用范围请读者查阅帮助。

当用户希望通过自建模型生成代码并在实时计算系统中运行这些代码时，用户就应该选择固定步长解法器来仿真模型。这是因为实时计算系统以固定的采样速率运行，若采用可变步长将有可能使仿真发生错误。

当用户并不从模型生成代码时，解法器类型的选择决定于模型的动态特性。当模型的状态变化特别快或是包含有不连续状态时，选择可变步长解法器可以缩短仿真时间。这是因为可变步长相对于固定步长的解法器需要更少的时间步，而两者仿真的精度相当。

关于该面板中其他参数的设置，读者可以查看在线帮助，它们都比较容易理解。

（2）【Data Import/Export 面板】

该面板主要用于向 MATLAB 7.0 工作空间输出模型仿真结果数据，或从 MATLAB 7.0 工作空间读入数据到模型，如图 7-62 所示。

【Load from workspace】：从 MATLAB 7.0 工作空间向模型导入数据，作为输入和系统的初始状态。

【Save to workspace】：向 MATLAB 7.0 工作空间输出仿真时间、系统状态、系统输出和系统最终状态。

【Save options】：向 MATLAB 7.0 工作空间输出数据的数据格式、数据量、存储数据的变量名以及生成附加输出信号数据等。

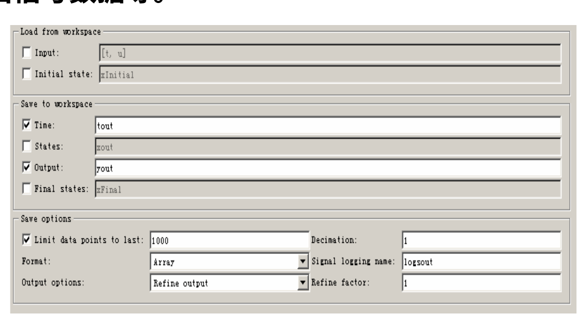


图 7-62　Data Import/Export 面板

（3）【Optimization 面板】

该面板用于设置各种选项来提高仿真性能和由模型生成的代码的性能，如图 7-63 所示。

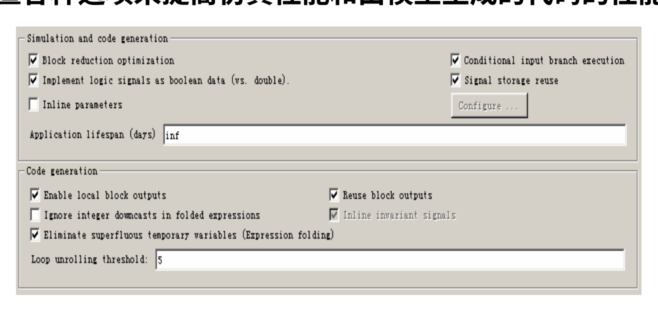


图 7-63　Optimization 面板

“Block reduction optimization”选项：设置用时钟同步模块来代替一组模块，以加速模型的运行。

“Conditional input branch execution”选项：用于优化模型的仿真和代码的生成。

“Inline parameters”选项：选中该选项使得模型的所有参数在仿真过程中不可调，Simulink 在仿真时就会将那些输出仅决定于模块参数的模块从仿真环中移出，以加快仿真。如果用户要想使某些变量参数可调，那么可以单击“configure”按钮打开“Model Parameter Configuration”对话框将这些变量设置为全局变量。

“Implement logic signals as boolean data (vs. double)”选项：使得接受布尔值输入的模块只能接受布尔类型，若该项没被选，则接受布尔输入的模型也能接受 double 类型输入。

（4）【Diagnostics 面板】

该面板主要用于设置当模块在编译和仿真遇到突发情况时，Simulink 将采用哪种诊断动作，如图 7-64 所示。该面板还将各种突发情况的出现原因分类列出，各类突发情况的诊断办法的设置在此不作详细介绍。

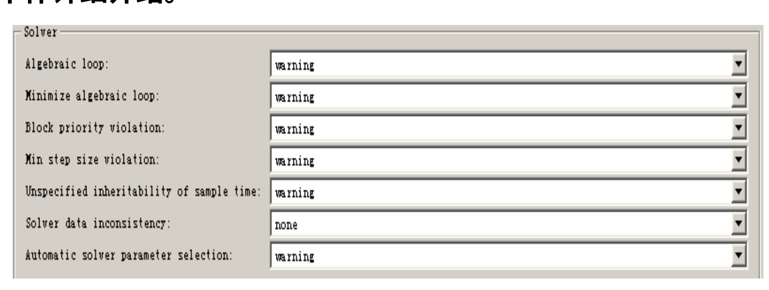


图 7-64　Diagnostics 面板

（5）【Hardware Implementation 面板】

该面板（如图 7-65 所示）主要用于定义硬件的特性（包括硬件支持的字长等），这里的硬件是指将来要用来运行模型的物理硬件。这些设置可以帮助用户在模型实际运行目标系统（硬件）之前通过仿真检测到以后在目标系统上运行可能会出现的问题，如溢出问题等。

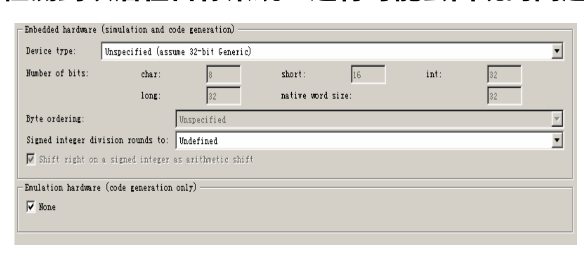


图 7-65　Hardware Implementation 面板

（6）【Model Referencing 面板】

该面板主要用于生成目标代码、建立仿真以及定义当此模型中包含其他模型或其他模型引用该模型时的一些选项参数值，如图 7-66 所示。

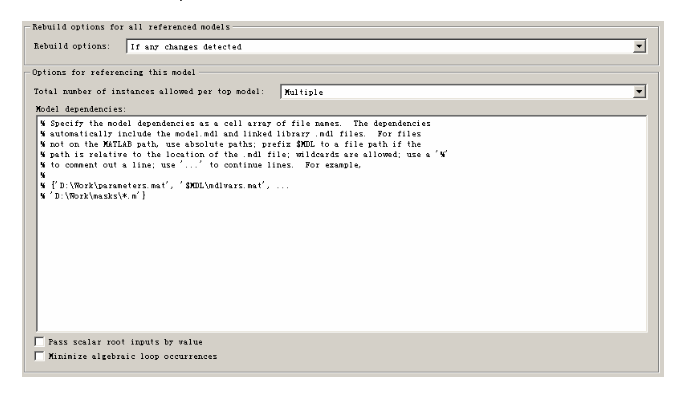


图 7-66　Model Referencing 面板

① 当前模型中含有其他模型时（Rebuild options for all referenced models 其他）的设置。

【Rebuild options】：用于设置是否要在当前模型更新、运行仿真和生成代码之前重建仿真和 Real-Time Workshop 目标。因为在进行模型更新、运行仿真和生成代码时，有可能其中所包含的其他模型发生了改变，所以需要在这里进行设置。

② 其他模型中包含有当前模型时（Options for referencing this model 其他）的设置。

【Total number of instances allowed per top model】：用于设置在其他模型中可以引用多少个该模型。

【Model dependencies】：用于定义存放初始化模型参数的命令以及为模型提供数据的文件名（通常是 MAT 文件或 M 文件）或文件的路径，定义的方法是将文件名或文件路径的字符串定义成字符串细胞阵列，如{'D:\Work\parameters.mat', '$MDL\mdlvars.mat', ... 'D:\Work\masks*.m'}。

【Pass scalar root inputs by value】：选中此项后，别的模型在调用该模型时就会通过数值来传递该模型的标量输入，否则就通过参考（如输入的地址）来传递输入。选中此项就会允许模型从速度快的寄存器或局部存储单元读取数据，而不是从它的实际输入位置来读取。如果模型的输入在一个时间步内发生改变，那么选中此项就会导致仿真出错。例如，当模型的输入/输出共享同一个局部存储单元（代数环）并且在一个时间步被调用多次时，模型只有当输入是通过参考传递时才会看到标量输入在同一时间步内发生的改变。

【Minimize algebraic loop occurrences】：选中此项后，Simulink 就试图消除模型中的一些代数环。

> 注意：所有通过仿真参数对话框设置的参数均可以通过 sim 和 simset 命令来设置。

2．仿真诊断对话框

如果 Simulink 在仿真运行过程中遇到错误，那么 Simulink 将停止仿真，并弹出仿真诊断对话框。通过该对话框，用户可以方便地在模型中找到出错的地方和出错的原因。将第 7.6 节中图 7-58 示例的 M 文件 ini_con 中的内容改为 con =[2 3]后再运行仿真，将弹出如图 7-67 所示的错误提示对话框。下面就介绍一下该对话框各部分提示的含义。

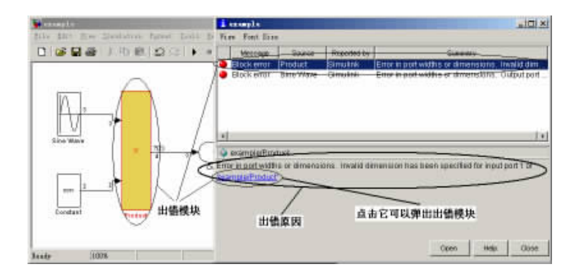


图 7-67　仿真诊断对话框

对话框分为上下两个面板，上面的面板列出了每一个错误的信息，这些信息包括：

- Message：消息类型（模块错误、警告、log 等）；
- Source：导致错误的模型元素（如模块或连线等）的名称；
- Reported by：报告错误的组件（如 simulink、Real-Time Workshop 等）；
- Summary：错误消息的简写，便于在列表中显示。

下面的面板显示当前所选中的 Message 的完整内容，包括出错原因和出错的元素。当用户选中某一个 Message 时，Simulink 就会打开模型窗口，并且将产生错误的模型元素用黄色加亮显示。

7.7.3　使用 MATLAB 命令运行仿真

使用模型窗口直接进行系统仿真时，每一次运行都是针对某一固定的模块参数设置，在运行过程中不能轻易地改变它。MATLAB 允许从命令行运行仿真，通过 MATLAB 命令进行模型的仿真使得用户可以从 M 文件来运行仿真。这样就允许不断地改变模块参数并运行仿真，也就让用户可以随机地改变参数和循环地运行仿真，用户就可以进行蒙特卡罗分析了。

MATLAB 7.0 提供了 sim 命令运行仿真，该命令完整的语法如下：

```
[t,x,y] = sim(model, timespan, options, ut);
```

或者输入下面的代码：

```
[t,x,y1, y2, ..., yn] = sim(model,timespan,options,ut);
```

其中只有 model 参数是必需的，其他的参量都被允许设置为空矩阵（[]）。Sim 命令中没有设置的或是设为空矩阵的参数的值等于建立模型时通过模块参数对话框设置的值或是系统默认的值；sim 命令中设置的参数值会覆盖模型建立时设置的参数值。如果你仿真的模型是连续系统，那么命令中还必须通过 simset 命令设定 solver 参数，默认的 solver 参数是求解离散模型的 VariableStepDiscrete。sim 中各参量的含义如表 7-25 所示。

表 7-25　　sim 命令参量

参量名	参量含义
T	返回仿真时间
X	返回仿真的状态矩阵
Y	返回仿真输出矩阵
Y1,…,yn	每一个 Yi 对应一个输出模块
model	模型名
Timespan	设置仿真的开始和结束时间
Options	用于设置仿真的相关参数的一个结构
Ut	模型输入

Simset 命令用于创建一个 options 结构，该结构中设定仿真参数和求解器的属性值。该命令的语法如下：

```
options = simset(property, value, ...);
options = simset(old_opstruct, property, value, ...);
options = simset(old_opstruct, new_opstruct);
simset
```

由于一般情况下，用户不会用到该命令，在此不作详细介绍，如果碰到可以查阅在线帮助，相关的命令还有 simplot、simget、set_param 等。用 sim 命令仿真第 7.6 节的示例，代码如下：

- 先在 MATLAB 7.0 命令窗口执行：

```
set_param('example','PreLoadFcn','ini_con');
set_param('example','StopFcn','out_plot');
```

- 然后执行：

```
[t,x,y]=sim('example')
```

使用 sim 命令运行仿真的结果如图 7-68 所示。

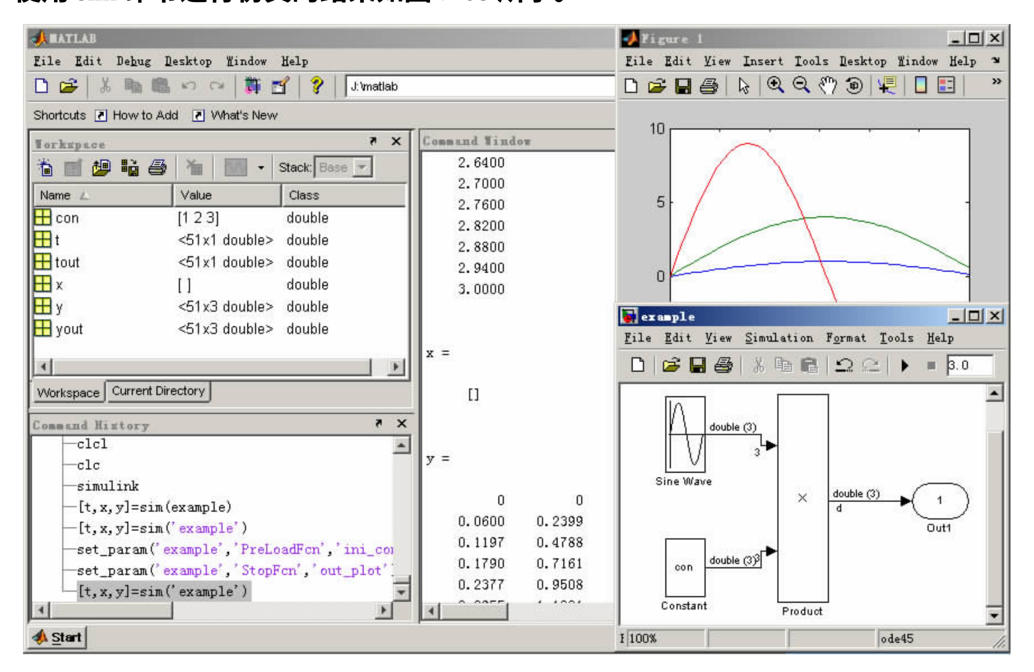


图 7-68 使用 sim 命令运行仿真

另外还可以通过 set_param 命令来开始、结束、暂停或者继续仿真，该命令的使用格式如下：

```
set_param('sys', 'SimulationCommand', 'cmd')
```

其中，'sys' 是要仿真的系统的名称；'cmd'是控制命令，可以取值 start、stop、pause、continue 或 update。类似的命令有 get_param，它可用来检查仿真的状态，其使用格式是：

```
get_param('sys', 'SimulationStatus')
```

该命令的返回值是 stopped、initializing、running、paused、updating、terminating 或者 external。

7.7.4 改善仿真性能及精度

Simulink 的仿真性能和精度受许多因素的影响，包括模型的设计和仿真参数的设置。对于大多数的问题，使用默认的仿真参数和解法器就可以得到满意的仿真结果。但是对于某些模型，适当的调整仿真参数可以得到更好的结果。

1．加速仿真

如果一个模型的仿真速度过慢，可能造成的原因有以下几种。

- 模型中有 MATLAB Fcn 模块。如果一个模型中含有 MATLAB Fcn 模块，那么在仿真时，Simulink 在每一个仿真时间步都要调用一次 MATLAB 解释器，所以应该尽可能

的使用 Simulink 的内置 Fcn 模块或是 Math Function 模块。

- 模型中有 M 文件的 S-函数。M 文件的 S-函数也会使每一个仿真步调用 MATLAB 解释器。这时应该将它转换成一个子系统或是 C-MEX 文件的 S-函数。
- 模型中含存储模块。使用存储模块将使阶数可变的解法器在每个时间步被重置为 1 阶。
- 仿真的时间步长的值太小。可将它设为默认值。
- 仿真精度设置要求太高。一般情况下，使用默认的 0.1%相对误差就可以。如果该值设得太小，那么这会使仿真在接近零状态附近时耗费更多的仿真步。
- 仿真时间设得太长。可适当减小时间间隔。
- 模型所求解的问题是一个 stiff 问题，而用户选择的是一个非 stiff 求解器。可以试用 ode15s。
- 模型包含有代数环。代数环是通过迭代的方法进行求解的，故仿真速度会比较慢。
- 模型中把一个 Random Number 模块作为 Integrator 模块的输入。若用户建立的是一个连续系统，可以使用 Sources 库中的 Band-Limited White Noise 模块作为 Integrator 模块的输入。
- 模型的采样时间彼此之间不是整数倍，求解器会选择足够小的时间步长来确保所有设置的采样点都能取到，因此会导致仿真速度下降。

2．改善仿真的精度

用户可以通过对比在不同的相对误差或绝对误差参数值时的仿真结果来判断解是否收敛。如果仿真的结果变化不大，则说明收敛。用户可以通过设置较小的仿真步长来保证仿真不会错过模型的关键行为。

如果仿真结果看起来不是很精确，那么有可能是下面的原因造成的。

- 模型含有取值接近零的状态。此时，如果绝对误差参数的值设置过大，那么仿真在接近零状态时仿真的时间步就太少。用户可以通过设置较小的绝对误差参数来解决这个问题。
- 如果减小绝对误差的办法还不能提高仿真精度，那么用户就应该减小相对误差参数的值以降低可容忍的误差和减小仿真的步长。

7.8　仿真结果分析

在第 7.1.4 节中讲过，仿真结果的可视化是 Simulink 建模的一个特点。不仅如此，Simulink 还可以对仿真的结果进行分析。

7.8.1　观看输出结果

在 Simulink 中有 3 种方法绘制模型仿真的输出：

- 在模型中将信号输入到 Scope 模块或是 XY Graph 模型；
- 将输出写入变量，然后使用 MATLAB 绘图命令；
- 将输出写入 To Workspace 模块，然后使用 MATLAB 绘图命令。

7.8.2 线性化

1．何为线性化

线性化就是将用户所建模型用状态空间矩阵形式 *A*、*B*、*C* 和 *D* 表示的线性方程模型。状态空间矩阵按如下方程描述线性的输入/输出关系：

$$x' = Ax + Bu$$
$$y = Cx + Du$$

其中，*x*，*u*，*y* 分别表示状态、输入和输出的向量。模型中的输入/输出必须使用 Simulink 提供的输入（In1）输出（Out1）模块。关于何为状态向量以及如何将一个系统用状态空间矩阵来描述等内容，请读者参考与信号与系统相关的书籍。

2．为何线性化

一旦数据具有状态空间的形式或是被转化成 LTI 对象，用户就可以使用控制系统工具箱里的函数作进一步的分析。

将状态空间转换为 LTI 对象：

```
sys = ss(A,B,C,D)
```

绘制系统的波特相位幅频图：

```
bode(A,B,C,D)  或 bode(sys)
```

求系统的时间响应：

```
step(A,B,C,D)  或  step(sys)：单位阶跃响应
impulse(A,B,C,D)  或  impulse(sys)：单位脉冲响应
lsim(A,B,C,D,u,t)  或 lsim(sys,u,t)：任意输入响应
```

3．如何线性化

用户可以通过下面几个函数对所建模型进行线性化：

```
[A,B,C,D] = linmod('sys');
[A,B,C,D] = linmod('sys',x,u);
[A,B,C,D] = linmod('sys', x, u, para);
[A,B,C,D] = linmod('sys', x, u, 'v5', para);
[A,B,C,D] = linmod('sys', x, u, 'v5', para, xpert, upert);
[A,B,C,D] = dlinmod('sys', x, u);
[A,B,C,D] = dlinmod('sys',Ts, x, u, 'v5', para);
[A,B,C,D] = dlinmod('sys', x, u, 'v5', para, xpert, upert);
argout = linmod2('sys', x, u, para);
```

linmod 用于线性化本身是线性和非线性的模型，dlinmod 用于线性化本身是离散系统或者离散和连续的混合系统。以上函数的具体使用方法及其中各个参数的含义，用户可以查询在线帮助。

7.8.3　平衡点的分析

Simulink 通过 trim 命令来决定动态系统的稳定状态点，所谓稳定状态点就是满足用户自定义的输入、输出和状态条件的点。下面通过一个简单的示例来介绍如何使用 trim 命令求解平衡点。

- 建立如图 7-69 所示的模型，保存为"exampbalance"；
- 为系统状态和输入定义一个初步的猜测值，并把用户预期的输出值赋给 y，代码设置如下：

```
>>x = [0;0;0];
>>u = 0;
>>y = [3;3];
```

- 使用索引变量规定模型的输入、输出和状态中哪些可变，哪些不可变，代码设置如下：

```
>>ix = [];
>>iu = [];
>>iy = [1;2];
```

- 使用 trim 命令来求出平衡点，代码设置如下：

```
>>[x,u,y,dx] = trim('exampbalance',x,u,y,ix,iu,iy)
```

- 得到如下结果：

```
x =
0.4000
0.0000
1.1314
u =
1.2000
y =
1.6000
0.4000
dx =
1.0e-023 *
0
0
0.1031
```

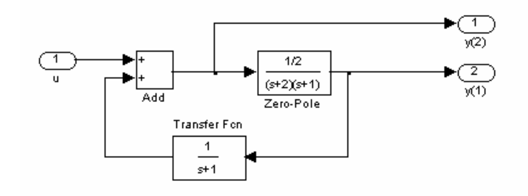


图 7-69　求解系统平衡点

注意：并不是所有求解平衡点的问题都有解。 如果无解，trim 将返回一个离期望状态的偏差最小的一个解。trim 命令还有其他的调用格式，请读者查阅在线帮助。

7.9 模型的调试

与许多高级编程语言一样，Simulink 6.0 也提供了一个调试器供用户查找和诊断模型中的错误，它允许用户通过单步运行（逐个方法地运行）仿真和显示模块的即时状态、输入和输出来查明错误。在 Simulink 6.0 中，用户既可以通过图形化的调试界面进行调试，也可以通过命令行进行调试。

7.9.1 Simulink 调试器

本小节通过一个示例简单地向读者介绍如何使用调试器。该示例是采取单步运行仿真的方式，用户可以通过这种运行方式来查看在每一个仿真步中各个模块的运行情况，从而可以检查模块功能是否与用户的设想一致。

- 打开第 7.3.2 节中如图 7-41 所示的示例。
- 选择 Tools→Simulink Debugger 菜单打开如图 7-70 所示的调试器，表 7-26 对该对话框工具栏各按钮的功能进行了详细介绍。

表 7-26　　调试器工具栏

工具栏按扭	功能
	进入当前方法
	跳过当前方法
	跳出当前方法
	在下一个仿真时间步跳转到第一个方法
	跳转到下一个模块方法
	开始或继续调试
	暂停仿真
	停止仿真
	在运行到下一个模块前跳出
	当选中的模块被执行时显示其输入输出
	显示选中的模块的当前输入输出
	选择动画模式
	显示调试器的帮助
Close	关闭调试器

调试器窗口界面中的各项设置的含义如下：

“points”页：用于设置断点，即仿真运行到某个模块方法或满足某个条件就停止；

“Simulation Loop”页：包含 Method、Breakpoints 和 ID 三列内容。用于显示当前仿真步正在运行的相关信息；

“Outputs”页：用于显示调试结果，包括调试命令提示、当前运行模块的输入、输出和模块的状态。如果是采用命令行调试，这些结果在 MATLAB 7.0 命令窗口中也会显示。调试命令提示显示当前仿真时间、仿真名和方法的索引号；

"Sorted List"页：用于显示被调试的模块列表，该列表按模块执行的顺序排列；

"Status"页：用于显示调试器各种选项设置的值以及其他状态信息。

- 单击按钮开始调试，在"Simulation Loop"页将显示当前运行的方法的名字，并且该方法也将显示在模块窗口中，如图 7-71 所示。当调试开始后，MATLAB 的命令窗口也会进入调试状态，如图 7-72 所示。用户可以在命令窗口采取命令行方式进行调试。
- 单击按钮进行仿真，如图 7-73 所示。在仿真过程中还可以设置断点（非条件中断或条件中断）；既可以按模块单步执行，也可以按时间步单步执行，还可以显示模型的信息等设置，用户可以查阅在线帮助，在此不做详细地介绍。

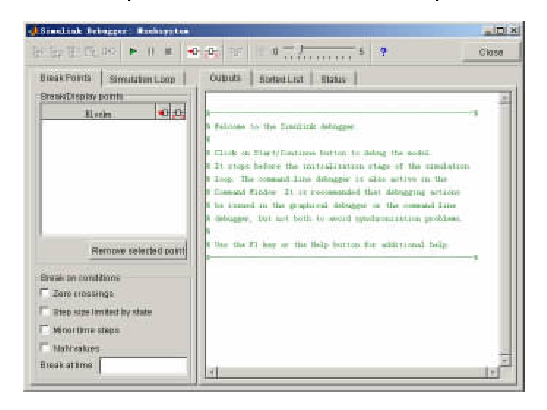


图 7-70　调试器窗口

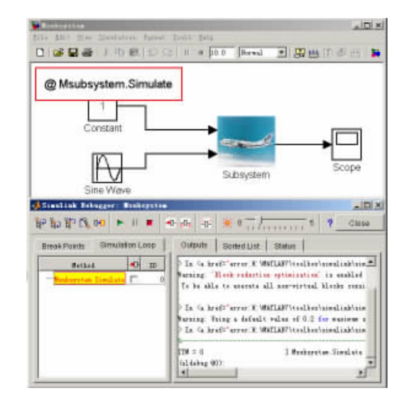


图 7-71　开始仿真

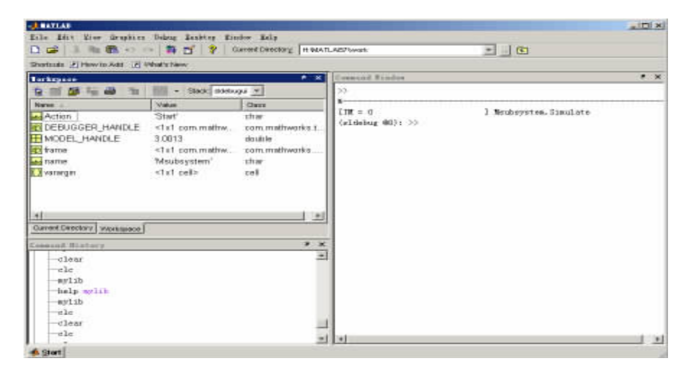


图 7-72　调试状态下的命令窗口

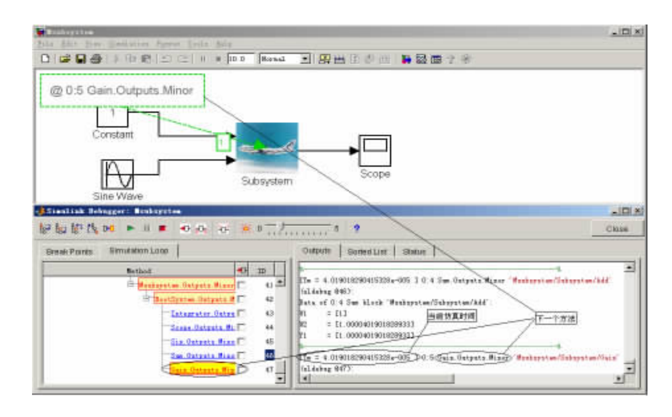


图 7-73　单步仿真

7.9.2 命令行调试

在命令行调试模式下，用户可以在 MATLAB 7.0 命令窗口中通过相应的命令来控制调试器。由于命令行调试方式需要用户记住许多的命令函数，所以一般用户可以直接通过调试用户界面的相关菜单和工具栏来操作，这里只向读者简单地介绍对于理解命令行调试很关键的几个基本概念，在后面地章节将陆续介绍一些常用的调试命令。

调试器接受缩写的调试命令，关于这些命令请读者查阅帮助文件。用户还可以通过在 MATLAB 7.0 命令行键入 Enter 键来重复执行前一命令。许多 Simulink 命令和消息是通过 Method ID 和 Block ID 来引用方法和模块的，Method ID 是按方法被调用的顺序从 0 开始分配的一个整数；Block ID 是在仿真的编译阶段分配的，形式为 sid:bid，sid 是一个用来表示系统的一个整数，bid 是模块在系统中的位置。用户可以通过 slist 命令来查看当前运行模型的每一个模块的 ID。

用户可以通过下面两个命令来启动调试器：

```
sim('vdp',[0,10],simset('debug','on'))
sldebug 'vdp'
```

其中的 vdp 是模型名。

7.9.3 设置断点

所谓断点就是指仿真运行到此处会停止仿真的地方，当仿真遇到断点停止时，用户可以使用 continue 命令跳过当前断点继续运行到下一个断点。调试器允许用户定义两种断点，即无条件断点和有条件断点，所谓无条件断点是指运行到此处就停止，而条件断点是指当仿真过程中满足用户定义的条件时才停止仿真。

如果用户知道自己的程序中某一点或当某一条件满足时就会出错，那么设置断点将很有用。下面向读者介绍两种设置断点的方法。

1．设置无条件断点

有 3 种方式设置。

- 通过调试器的工具栏

先在模型窗口中选择要设置断点的模块，然后用鼠标左键单击按钮。在如图 7-74 所示的示例模型中的 Scope 模块处设置断点，可用“Remove selected point”按钮删除已设置好的断点。

- 通过调试器的 Simulation Loop 页

选择该页的 Breakpoints 列中要设置的断点处选择前面的选择框即可。

- 通过在 MATLAB7.0 命令窗口运行相关命令

使用 break 和 bafter 命令可以分别在一个方法的前面和后面设置断点，使用 clear 命令清除断点。

break 命令的语法，代码如下：

```
break
break m:mid      // mid 为方法 ID
```

```
break <sid:bid | gcb> [mth] [tid:TID]    // sid:bid 为模块 ID，gcb 为当前选中的模块，TID 为任务 ID，mth 为方法名，如：Outputs.Major
break <s:sid | gcs> [mth] [tid:TID]    // sid 为系统 ID，gcs 为当前选中的系统
break mdl [mth] [tid:TID]    // mdl 为当前选中的模型
```

bafter 和 clear 的语法与之类似，在此不作详细介绍，请读者查阅帮助。

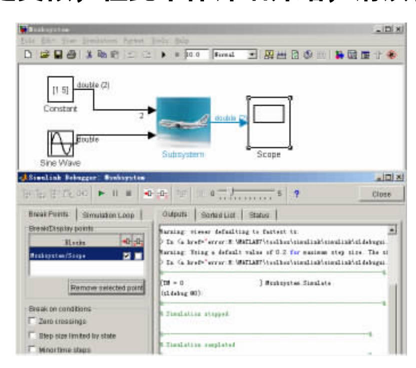


图 7-74　通过工具栏设置断点

2．设置有条件断点

设置有条件断点可以通过在调试器"Break on conditions"页中设置相应的断点条件来实现，也可以通过以下命令来设置：

```
tbreak：在仿真时间步处
minor：在引起最小时间步的方法处
nanbreak：当发生上溢或下溢时
xbreak：当仿真到决定时间步长的状态时
zcbreak：当在一个时间步内发生过零时
```

7.9.4　显示仿真的有关信息

Simulink 调试器提供了许多命令允许用户在模型运行时显示模块的状态、模块的输入/输出和其他的信息。在这里主要介绍如何使用调试器工具条中的按钮来执行相应的信息显示，当然这些按钮也有相应的命令，在此不详细介绍。

按钮用于显示模块的输入/输出信息。首先在模型窗口选中模块，然后用鼠标左键单击该按钮，被选中的模块在当前仿真步的输入、输出和状态信息将显示在调试器窗口的"Outputs"页中，相应的命令有 probe 和 disp。在如图 7-75 所示的示例模型中进行单步调试（方法按第 7.9.1 节介绍的方法进行）到下面的时间步

```
[Tm = 1.205705487124599e-005 ] 0:5 Gain.Outputs.Minor 'Msubsystem/Subsystem/Gain'
(sldebug @64):
```

然后就可以看到图中的信息，如图 7-75 所示。使用相应的 probe 命令来显示模块信息，probe 命令的语法如下：

```
probe：进入或退出 probe 模式，当进入时用于显示选中模块的输入输出，如图 7-75 所示。
probe gcb：显示选中模块的当前输入输出；
```

probe s:b：s 为系统的序号，b 为模块编号，如图 7-76 所示。0:4 对应示例 Msubsystem 系统中 Subsystem 子系统的 Add 模块

disp 命令可实现相同的功能，其使用语法与 probe 类似。

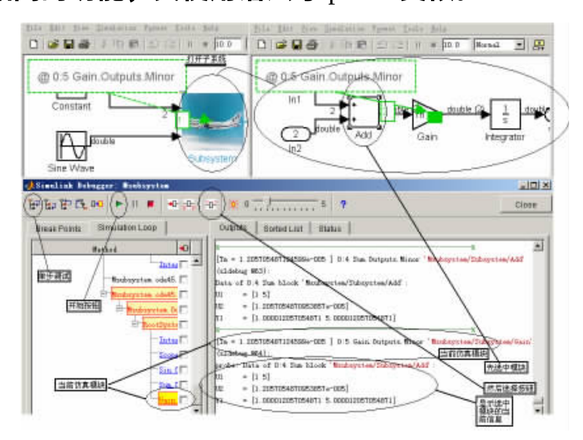


图 7-75　通过按钮显示模块信息

按钮也是用于显示模块的信息，与按钮不同的是，如果用户为某一模块设置了断点，那么用户就可以在“Break points”页的 Break/Display points 面板中选中列的相应模块的选框，一旦为某一模块设置了该选框，那么仿真在每一次运行到该模块处时就会显示它的相关信息。对应的可以分别使用 trace 和 untrace 命令来完成同样的功能设置和撤销设置。atrace 命令用于显示代数环的信息，其语法及含义如表 7-27 所示。

states 命令用于在 MATLAB 7.0 命令窗口中显示系统的当前信息，如对图 7-75 中的示例运行该命令即可得到：

```
(sldebug @64): >> states
Continuous States:
Idx   Value                      (system:block:element   Name     'BlockName')
   0   1.205712755753207e-005    (0:0:0   CSTATE   'Msubsystem/Subsystem/Integrator')
   1. 6.0285347042516e-005       (0:0:1)
```

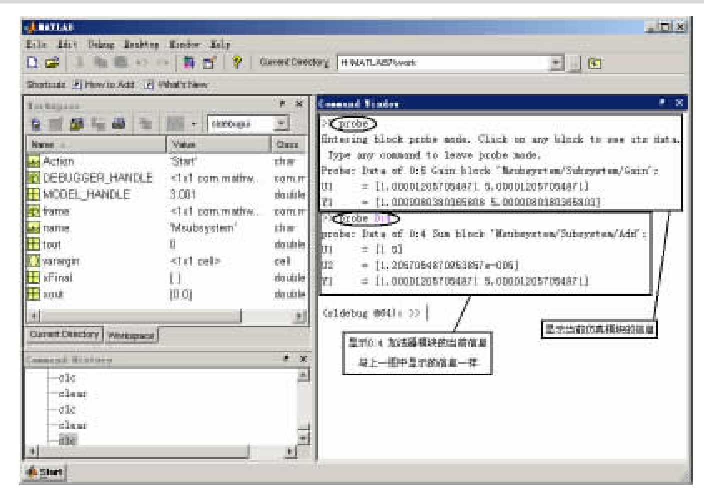


图 7-76　通过 probe 命令显示模块信息

ishow 命令用于锁定积分信息的显示。当它被使能时，那么仿真时间步每前进一次或遇到需要改变仿真步长的状态时，调试器都将显示一个相应的消息。

表 7-27　　trace 命令

命令	显示的信息
atrace 0	没有信息显示
atrace 1	显示环路变量的解，求解环路所需的迭代次数，以及估计的解的误差
atrace 2	与 atrace 1 相同
atrace 3	除了 atrace2 所显示内容外，还显示求解环路的 Jacobian 矩阵
atrace 4	除了 atrace 3 所显示内容外，还显示环路变量在每次迭代时的瞬时解

7.9.5　显示模型的信息

调试器除了可显示关于仿真的信息外，还可显示有关模型的信息。

slist 命令用于显示系统中各模块的索引，模块的索引其实也是按它们执行的顺序来排列的，对图 7-75 中的示例运行 slist 命令将看到如下信息：

```
(sldebug @64): >> slist
---- Sorted list for 'Msubsystem' [8 nonvirtual blocks, directFeed=0]
   0:0     'Msubsystem/Subsystem/Integrator' (Integrator, tid=0)
   0:1     'Msubsystem/Scope' (Scope, tid=0)
   0:2     'Msubsystem/Constant' (Constant, tid=1)
   0:3     'Msubsystem/Sine Wave' (Sin, tid=0)
   0:4     'Msubsystem/Subsystem/Add' (Sum, tid=0)
   0:5     'Msubsystem/Subsystem/Gain' (Gain, tid=0)
   0:6     'Msubsystem/Floating Scope1' (Scope, tid=1)
   0:7     'Msubsystem/Scope1' (SignalViewerScope, tid=0)
```

这与调试器窗口中“Sorted List”页显示的内容一样。

其他用于显示模块信息的命令还有 bshow、systems、zclist、ashow、status 等，它们的使用方法与前面的命令也非常类似，具体的操作细节和所实现的功能请读者查阅帮助。

7.10　S-函数

S-函数（System-函数）为扩展 Simulink 的功能提供一个强有力的机制，本小节将向读者解释以下几个问题：

- S-函数是什么？
- 用户为什么要使用 S-函数？
- 用户怎样书写自己的 S-函数？

7.10.1　什么是 S-函数

S-函数是一种描述动态系统的计算机语言，用户可以用 MATLAB、C、C++、Ada 或 Fortran 语言书写。C、C++ 、Ada 和 Fortran 书写的 S-函数用 mex 命令编译成 MEX 文件，它们就像 MATLAB 中其他的 MEX 文件一样可以动态的连接到 MATLAB。本文只介绍用 MATLAB 语言书写的 S-函数。

S-函数采用一种特殊的调用语法使得函数可以和 Simulink 方程解法器进行交互，这种交互与解法器和 Simulink 系统自身提供的模块间的交互十分类似。S-函数的形式十分通用，用户可以用它描述连续、离散和混合系统。

7.10.2　为何要用 S-函数

通过 S-函数，用户可以实现以下操作：

- 用户可以通过它用多种语言来创建新的通用性的 Simulink 模块；
- 只需遵循一系列简单的规则，用户即可将自己的算法用 S-函数实现。当用户写好自己的 S-函数后，可以在 User-Defined Functions 模块库中的 S-function 模块中通过名称来调用编写好的 S-函数，用户还可以封装它；
- 用户可以通过 S-函数将一个系统描述成一个数学方程；
- 便于使用图形化仿真；
- 可以创建代表硬件驱动的模块。

使用 S-函数的主要优点是用户可以通过它创建通用性模块，可以多次调用，而且可以每次使用不同的参数值。

7.10.3　S-函数如何工作

要理解 S-函数如何工作，读者应该先理解如何用数学描述一个模块以及 Simulink 如何仿真一个模型。

1．模块的输入、状态和输出间的数学关系

描述一个 Simulink 模块需要 3 个基本元素，即输入向量（u）、状态向量（x）和输出向量（y），输出是输入向量和采样时间的函数：

$$y = f_0(t,u,x) \quad \text{输出}$$
$$\dot{x}_c = f_d(t,x,u) \quad \text{微分}$$
$$x_{d_{k+1}} = f_u(t,x,u) \quad \text{更新}$$

$$x = x_c + x_d$$

Simulink 在仿真时把上面的这些方程对应为不同的仿真阶段，它们分别是计算模块的输出、更新模块的离散状态和计算连续状态的微分。在仿真的开始和结束，还包括初始化和结束任务两个阶段。在每一个阶段，Simulink 都重复地对模型进行调用。

2．Simulink 运行仿真的过程

仿真按照如图 7-77 所示的流程进行，由图可知仿真是分阶段进行的。在初始化阶段，Simulink 将库中的模块并入到用户自建模型中，确定模块端口的数据宽度、数据类型和采样时间，评估模块参数，决定模块运行的优先级，定位存储地址；然后进入仿真循环，每一次的仿真循环称为一个仿真步。在每一个仿真步中的不同仿真阶段，Simulink 按优先级运行模型中的每一个模块。如果在积分时对仿真的时间步长有要求，则此时需要将时间步细化；完成一个仿真循环（也即一个仿真步）后，就进入下一个循环仿真步，如此循环直至仿真结束。含有 S-函数模块的模型的仿真过程与此类似。

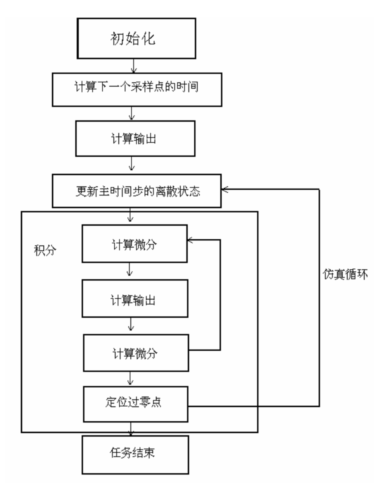


图 7-77　仿真执行流程图

3．S-函数的回调方法

一个 S-函数是由一系列 S-函数回调的方法组成，在每次仿真循环中的每一个仿真阶段由 Simulink 调用相应的 S-函数回调方法来执行相应的任务。至于 S-函数有哪些回调方法，以及 Simulink 如何确定在某一个仿真阶段调用哪一个回调方法，将在下一小节介绍。

与一般模型的仿真类似，S-函数回调方法用以完成以下任务：

- 初始化：在进入第一个仿真循环之前，Simulink 初始化 S-函数。在此阶段，Simulink 主要完成初始化 SimStruct（SimStruct 包含 S-函数信息的数据结构），确定输入输出端口的数目和大小，确定模块的采样时间，分配内存和 Sizes 数组的工作。
- 计算下一个采样点。如果模型使用变步长解法器，那么就需要在当前仿真步确定下一个采样点的时刻，即下一个仿真步的大小。
- 计算当前主仿真步的输出。当这次回调完成后，模块的所有输出端口的值对当前仿真时间步有效，即模块的输出被更新后才能作为其他模块的有效输入去影响那些模块的行为。
- 更新当前主时间步的离散状态。在此仿真阶段，所有的模块都要为当前时间的仿真循环更新离散状态。
- 积分。只有当模块具有连续状态或者非采样过零点时，Simulink 才会有这一仿真阶段。对于 S-函数，在该仿真阶段 Simulink 按最小时间步来调用 S-函数的输出和微分 S-函数方法。如果 S-函数具有非采样过零点，Simulink 按最小时间步来调用 S-函数中的输出和过零点部分，以便 Simulink 确定过零点的位置。

7.10.4 怎样书写 S-函数

下面将使用 MATLAB 语言来书写 S-函数称为 M 文件 S-函数，每一个 M 文件 S-函数包含一个如下形式的 M 函数：

```
[sys,x0,str,ts]=f(t,x,u,flag,p1,p2,...)
```

表 7-28 列出了上式中各参数的含义。在这类 S-函数中的 S-函数回调方法是用 M 文件子函数的形式来实现的。

表 7-28 函数各参数的含义

参数名	参数含义
f	S-函数的名称
t	当前仿真时间
x	S-函数模块的状态向量
u	S-函数模块输入
flag	用以标示 S-函数当前所处的仿真阶段，以便执行相应的子函数
p1,p2,…	S-函数模块的参数
ts	向 Simulink 返回一个包含采样时间和偏置值的两列矩阵。不同的采样时刻设置方法对应不同的矩阵值。如果希望 S-函数在每一个时间步都运行，就设为[0 0]；如果希望 S-函数模块与和它相连的模块以相同的速率运行，就设为[-1 0]；如果希望可变步长，则设为[2 0]；如果希望从 0.1s 开始每隔 0.25s 运行一次，就设为[0.25 0.1]；如果你的 S-函数执行多个任务，而每个任务运行的速率不同，可设为多维矩阵，两个任务设为[0.25 0; 1.0 0.1]…
sys	用以向 Simulink 返回仿真结果的变量。根据不同的 flag 值，sys 返回的值也不完全一样（因为不同的 flag 对应不同的仿真阶段和仿真任务，仿真也就得到不同的结果）
x0	用以向 Simulink 返回初始状态值
str	保留参数

在模型仿真过程中，Simulink 重复地调用 f，并根据 Simulink 所处的仿真阶段为 flag 参量传递不同的值，同时为 sys 变量指定不同的角色（不同的角色对应不同的返回值）。flag 用来标示 f 函数要执行的任务，以便 Simulink 调用相应的子函数（也即 S-函数的回调方法）。

因此，用户在编写 M 文件 S-函数时只需用 MATLAB 语言来为每个 flag 值对应的 S-函数方法编写代码即可。表 7-29 列出了在各个仿真阶段对应要执行的 S-函数回调方法以及相应的 flag 参数值。

表 7-29 各个仿真阶段对应要执行的 S-函数方法

仿真阶段及方法说明	S-函数方法	Flag
初始化。定义 S-函数模块的基本特性，包括采样时间、连续或离散状态的初始条件和 Sizes 数组	mdlInitializeSizes	flag = 0
计算下一个采样点的绝对时间。该方法只有在用户在 mdlInitializeSizes 说明了一个可变的离散采样时间时可用	mdlGetTimeOfNextVarHit	flag = 4
计算输出	mdlOutputs	flag = 3
更新离散状态	mdlUpdate	flag = 2
计算微分	mdlDerivatives	flag = 1
结束仿真	mdlTerminate	flag = 9

用户可以通过在 MATLAB 7.0 命令窗口输入“<<sfundemos”命令来查看 S-函数示例，如图 7-78 所示。其中提供了一个“M-file S-function Template”示例，它是为用户书写 S-函数提供的一个模板。使用模板编写 S-函数时，用户可将 S-函数名换成期望的函数名，若需要额

外的输入参数量，还需要在输入参数列表的后面增加这些参数。用户剩下的任务就是根据所编 S-函数要达到的目的，用相应的代码去代替模板里各个子函数的代码。

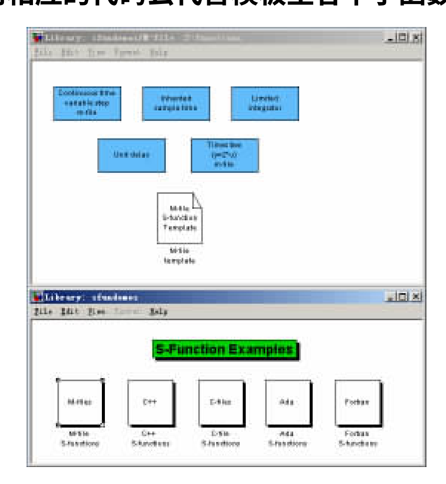


图 7-78　S-函数示例

“M-file S-function Template”中的“M-file S-function”模块的文件代码如下（删掉了其中的解释性文字）：

```
function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)


switch flag,
   case 0,
      [sys,x0,str,ts]=mdlInitializeSizes;
   case 1,
      sys=mdlDerivatives(t,x,u);
   case 2,
      sys=mdlUpdate(t,x,u);
   case 3,
      sys=mdlOutputs(t,x,u);
   case 4,
      sys=mdlGetTimeOfNextVarHit(t,x,u);
   case 9,
      sys=mdlTerminate(t,x,u);
   otherwise
      error(['Unhandled flag = ',num2str(flag)]);
end


function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates      = 0;
sizes.NumDiscStates      = 0;
```

```
sizes.NumOutputs          = 0;
sizes.NumInputs           = 0;
sizes.DirFeedthrough      = 1;
sizes.NumSampleTimes      = 1;
sys = simsizes(sizes);
x0    = [];
str = [];
ts    = [0 0];

function sys=mdlDerivatives(t,x,u)
sys = [];

function sys=mdlUpdate(t,x,u)
sys = [];

function sys=mdlOutputs(t,x,u)
sys = [];

function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;
sys = t + sampleTime;
function sys=mdlTerminate(t,x,u)
sys = [];
```

其中 mdlInitializeSizes 方法中的 sizes 是一个结构，它是 S-函数信息的载体，其各个字段的意义如表 7-30 所示。模板中 simsizes 函数的作用与用法可查阅帮助。

表 7-30 sizes 各字段的意义

字段名	含义
sizes.NumContStates	连续状态的个数
sizes.NumDiscStates	离散状态的个数
sizes.NumOutputs	输出的数目（所有输出向量的宽度之和）
sizes.NumInputs	输入的数目（所有输入向量的宽度之和）
sizes.DirFeedthrough	有无直接馈入
sizes.NumSampleTimes	采样时间的个数

S-函数模块还可实现直接馈入、输入信号宽度动态可变以及多种采样时间的设置，S-函数还可以用其他计算机高级语言书写，但其基本原理是一样的，鉴于篇幅有限，这些内容请读者自己查阅帮助。

7.10.5 S-函数应用示例

本小节只介绍如何利用‘User-Defined Functions’库中的 S-Function 模块创建由 MATLAB 语言书写的 M 文件 S-函数。对于如何使用其他语言和方法来书写，请读者查阅帮助。

用 S-函数实现一个传输函数为$[(S+1)(S+2)]^{-1}$的模块，如图 7-79 所示。

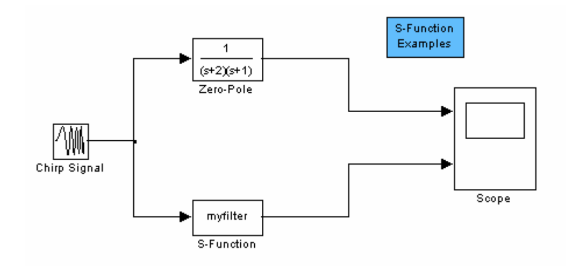


图 7-79　S 函数模块

其中 Zero-Pole 模块和 S-function 模块实现的功能一样，目的是检验所写的 S-函数模块是否正确。S-function Example 模块是为了方便打开 M-文件 S-函数书写的模板，通过模板书写好的 S-函数文件保存为 myfilter.m。S-function 模块的 S-函数名参数设为 myfilter，scope 模块的输入坐标轴数目设为 2，其他模块的参数采用默认值。关于如何将传输函数转换成状态方程，请查阅有关信号与系统的书籍。

S-函数文件内容如下：

```
function [sys,x0,str,ts] = myfilter(t,x,u,flag)
A=[0 1;
   -2 -3];
B=[0;
   1];
C=[1 0];
D=[0];
switch flag,
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D);
  case 1,
    sys=mdlDerivatives(t,x,u,A,B,C,D);
  case 2,
    sys=mdlUpdate(t,x,u);
  case 3,
    sys=mdlOutputs(t,x,u,A,B,C,D);
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
  case 9,
    sys=mdlTerminate(t,x,u);
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D)
```

```
sizes = simsizes;
sizes.NumContStates    = 2;
sizes.NumDiscStates    = 0;
sizes.NumOutputs       = 1;
sizes.NumInputs        = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0   = [0;
        0];
str = [];
ts   = [0 0];
function sys=mdlDerivatives(t,x,u,A,B,C,D)
sys = A*x+B*u;
function sys=mdlUpdate(t,x,u)
sys = [];
function sys=mdlOutputs(t,x,u,A,B,C,D)
sys = C*x+D*u;
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;
sys = t + sampleTime;
function sys=mdlTerminate(t,x,u)
sys = [];
```

仿真结果如图 7-80 所示，上下两个模块的输出结果一样，这就证明所写 S-函数功能正确。读者可以试着书写自己的 S-函数模块。

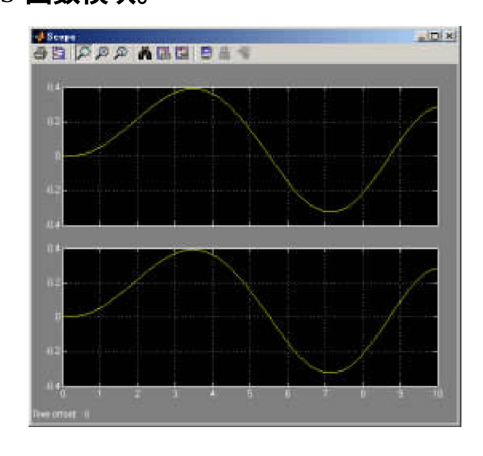


图 7-80　S-函数模型仿真结果

> **注意：S-函数保存的文件名和函数名要一致，保存 M-文件 S-函数的目录要包含在 MATLAB 7.0 的搜索路径文件之中。**

7.11　综合实例——PLL 中的非线性电荷泵和滤波器

在结束本章之前，将通过创建一个较为复杂的模型向读者总结如何利用 Simulink 来创建自己的模型并进行仿真。通过本小节的学习，读者将会对前面的内容有一个更为深刻的理解，对 Simulink 建模有一个较为全面的认识。

下面创建锁相环中电荷泵和滤波器部分的模型，系统示意图如图 7-81 所示。

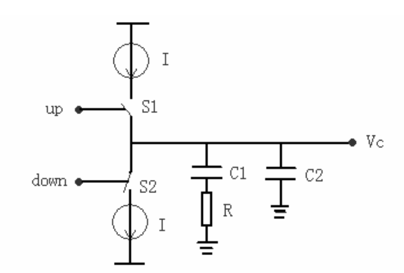


图 7-81　电荷泵和滤波器

下面按第 7.2.2 节介绍的建模步骤进行该系统的建模。

- 系统分析

创建该模块后，当 *up* 为高电平时电流 *I* 对电容充电，当 *down* 为高电平时电流 *I* 对电容放电，*up* 和 *down* 不会同时为高电平。C_1、C_2 和 *R* 组成的部分实现的是一个滤波器的功能，可以用一个 Transfer Fcn 模块来实现，其实现的传输函数是：

$$H_{(s)}=\frac{V_{(s)}}{I_{(s)}}=\frac{\frac{1}{C_2}(S+\frac{1}{R*C_1})}{S^2+\frac{S(C_1+C_2)}{R*C_1*C_2}}$$

其中 *R*、C_1 和 C_2 可设为变量，并将该模块进行封装。

本模型难以实现的是电荷泵部分。当 *up* 为高电平时，电流 I 以正阶跃信号的形式输入滤波器模块；当 *down* 为高电平时，电流 *I* 以负阶跃信号的形式输入滤波器模块，但是 Simulink 库中 Sources 子库提供的 Step 模块的阶跃产生时间参数难以设置成随 *up* 或 *down* 信号高电平到达时间而变。在此采用 User-Defined Functions 子库中的 Embedded MATLAB Funtion 模块实现一个电流阶跃的产生时间随 *up* 或 *down* 信号的到达时间而变的电荷泵。

- 启动 Simulink 模块库浏览器，创建如图 7-82 所示的电荷泵和滤波器模型。

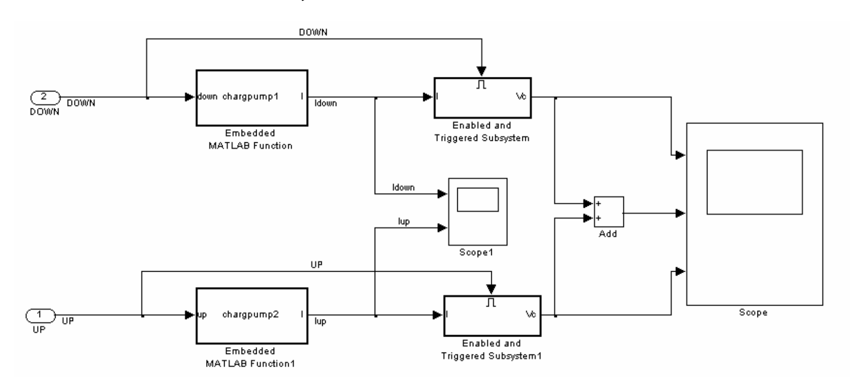


图 7-82　电荷泵和滤波器模型

Embedded MATLAB Function 模块中的文件如下：

```
function I = chargpump1(down)
if down>0
    I= -25e-6;    //当 down＞0 时，输出负电流，大小为-25uA，不同的电荷泵取值不一样
else
    I= 0;    //当 down≤0 时，输出 0 电流
end
```

Embedded MATLAB Function1 模块中的文件如下：

```
function I = chargpump2(up)
if up>0
    I=25e-6;   //当 up＞0 时，输出正电流，大小为 25uA，不同的电荷泵取值不一样
else
    I=0;    //当 up≤0 时，输出 0 电流
end
```

滤波器的功能由 Ports&Subsystem 子库中的使能子系统来实现，两个使能子系统模块的内部结构完全一样，如图 7-83 所示。

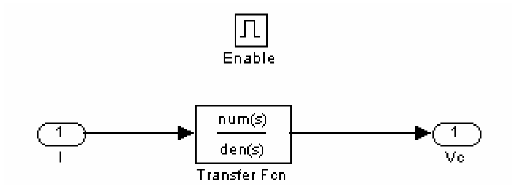


图 7-83　滤波器子系统

其中 Transfer Fcn 模块的参数设置如图 7-84 所示。Enable 模块的"States when enabling"参数设为 reset，若设为 held，那么滤波器（Transfer Fcn 模块）不会将输入电流的跳变当成一个阶跃信号来处理。这样每一次使能信号有效后，Transfer Fcn 模块的状态就会重置成初始状态，体现在输出上就是在每一次 Enable 信号有效时输出都是从 0 开始变化。这也是在系统模型中用两个滤波器来分别处理输入的正阶跃和负阶跃电流的原因。系统的最终输出是两个滤波器输出的和，因为 *up* 和 *down* 不会同时为高电平，所以这样做可以保证每一时刻的输出都能反映前面所有时间的输入对系统状态的积累效果。如果要考虑有 *up* 和 *down* 同时为高电平的情况，那么可以在 *up* 输入和 *down* 输入与 Embedded MATLAB Function 模块的输入端口之间加一模块，使得在此种情况下进入 Embedded MATLAB Function 模块 down 端口和 Embedded MATLAB Function1 模块 up 端口的信号值都为 0，即没有电流输入到滤波器，这与实际相吻合。

Enable 模块还有另外一个作用是保证滤波器只对使能信号上升边沿的阶跃信号进行处理，即只接收 *Idown* 从 0→-25 和 *Iup* 从 0→25 的阶跃信号，而不接收 *Idown* 从-25→0 和 *Iup* 从 25→0 的阶跃信号。读者还可以试着将子系统中的 Enable 模块换成 Trigger 模块，观察会出现什么问题。

如果读者对上面这些有关信号处理的知识不了解的话，那么你也可以不必知道为什么要这样设置，而只需学习模型创建的过程即可。

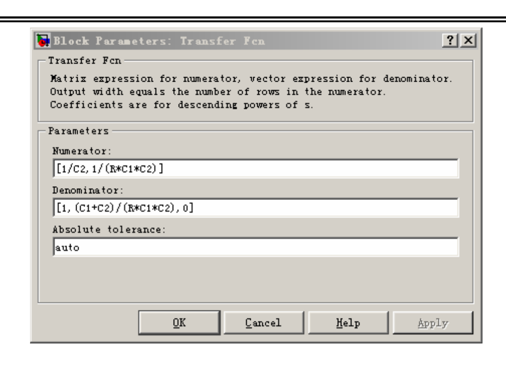


图 7-84　Transfer Fcn 模块的参数设置

- **封装子系统**

对由使能子系统构造的滤波器模块进行封装（方法参见第 7.3.2 节），其中封装的设置情况如图 7-85 所示。用鼠标左键双击封装好的使能子系统图标将会弹出如图 7-86 所示参数设置对话框，用户可在此对话框中设置子系统的参数。

- **设置参数**

在第二步中已对各主要模块的参数设置都进行了说明，下面主要设置与仿真有关的设置。主要需要设置的是仿真开始和结束的时间、解法器的类型和解法器、最大和最小仿真步长以及相对和绝对的仿真误差容忍度等，具体设置如图 7-87 所示。读者可以对其中的设置进行修改，然后进行仿真，观察输出因设置的不同而相应发生了哪些变化。

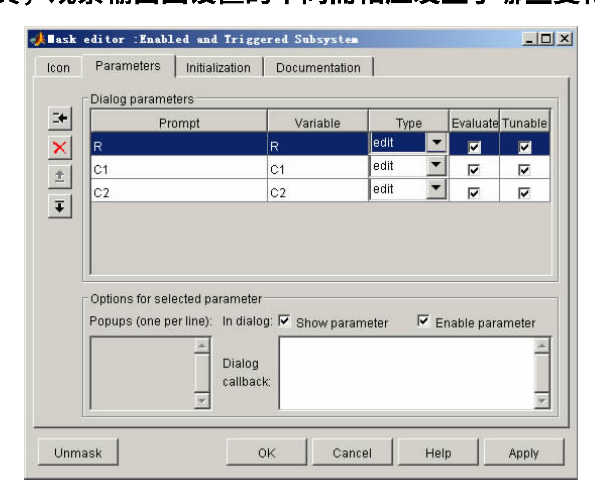


图 7-85　使能子系统的封装参数设置

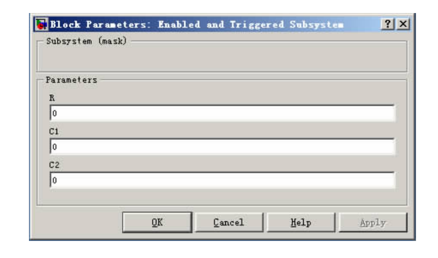


图 7-86　封装使能子系统的参数设置对话框

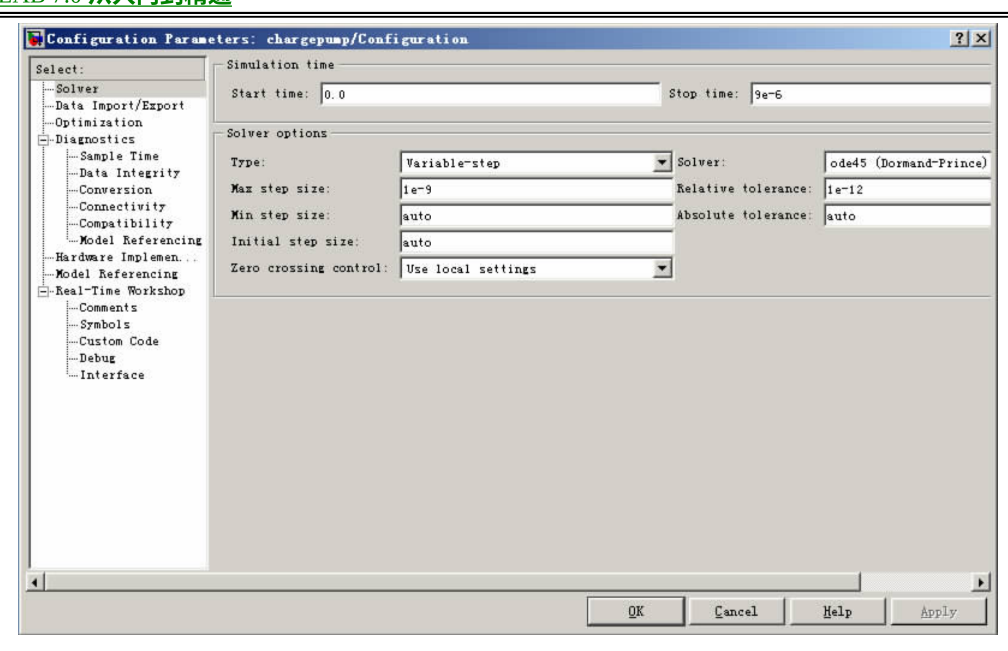


图 7-87　仿真参数设置

- **保存模型**

为了仿真方便，读者可以将模型中的 *up* 和 *down* 输入模块用一信号源模块（如 Pulse Generator、Sine Wave 等）代替，这样就不用通过 MATLAB 命令来为模型添加输入信号。如果仿真结果不够理想，那么可以调整模块参数和仿真参数的设置，直至仿真结果吻合与建模用户预想的情况相符。仿真结果如图 7-88 和、图 7-89 和图 7-90 所示，相关模块的参数设置如下：

滤波器子系统模块中 R=31.8e3，C_1=62.2e-12，C_2=6e-12；输入信号为脉冲信号，周期为 2e-6、幅度为 1、占空比 50% ，*up* 和 *down* 相位相差 180°；仿真参数的设置如图 7-87 所示。

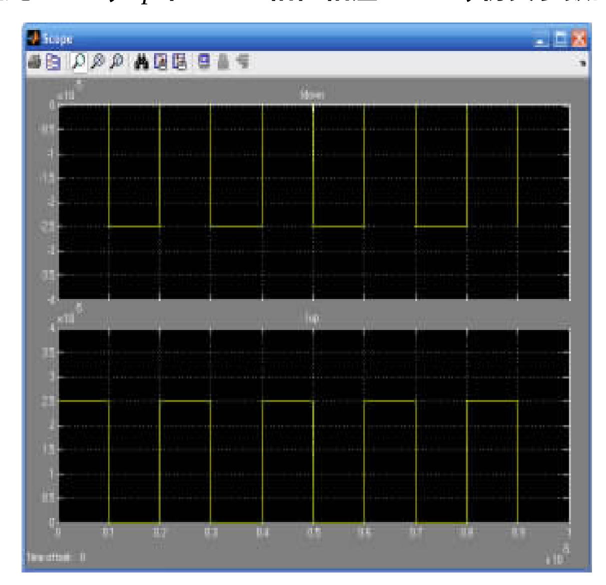


图 7-88　*Iup* 和 *Idown* 仿真结果

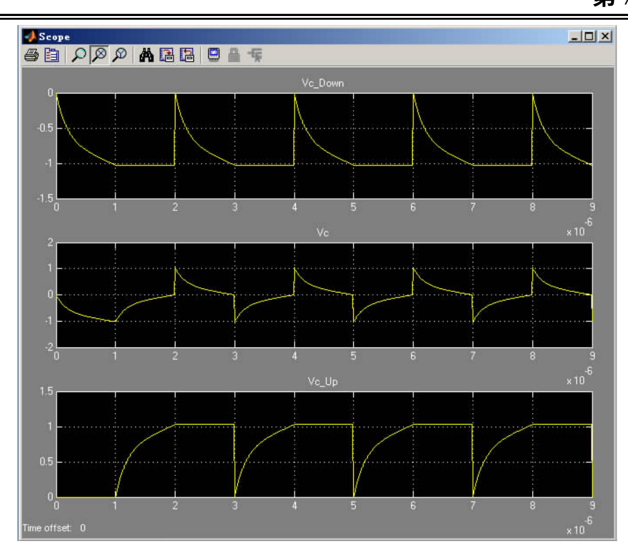


图 7-89　*Vc_Down*、*Vc* 和 *Vc_Up* 的仿真结果

由图 7-89 可以看出滤波器只对 *Iup* 的上升跳变和 *Idown* 的下降跳变有效，而且每当滤波器的使能信号有效时，它的输出都是从 0 开始变化。

若将模型中上面的一个滤波器子系统中 Enable 模块的“States when enabling”参数设为 held，那么仿真结果将如图 7-90 所示。对比图 7-89 和图 7-90，可以证明第二步中所述分析是正确的。改为 held 后，只有开始的一个电流跳变被当作阶跃信号来处理，其余的均当作常量来处理的，因为对应处的 *Vc_Down* 是线性变化的。

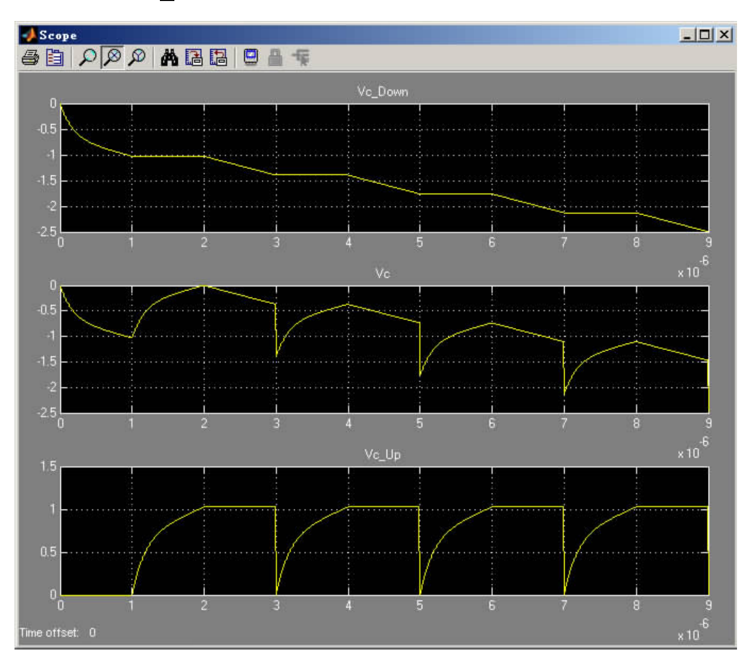


图 7-90　改变 Enable 模块的参数后的仿真结果

- 调试

有时用户对所建模型的输出结果并没有一个预想的值，因此只是简单的通过观察结果并

不能证明所建模型的正确性，这时就需要进行调试（方法参见第 7.9 节），以观察系统的每一个模块在每一个时间步的运行情况。对于上面的模型，可能会因为读者的操作系统和 MATLAB 7.0 的安装情况不同而有所不同，在此不作详细的解释。

本小节通过创建一个较为复杂的模型向读者展示了在实际的建模过程中，用户会经常碰到的某些问题。示例中遇到的最大问题是当 Simulink 库中没有完全满足用户要求的现成模块时，用户该如何去利用已有模块来构造需要的模块，以及如何验证模型的正确性。当然只通过一个示例不足以展示所有建模的细节问题，但是通过对本章内容的全面学习，相信读者将会知道如何去解决实际操作中碰到的各种具体问题。

第 8 章　MATLAB 7.0 的符号计算功能

符号数学工具箱中的工具是建立在功能强大的 Maple 软件的基础上。它最初是由加拿大的滑铁卢（Waterloo）大学开发出来的。如果要求 MATLAB 7.0 进行符号运算，那么首先由 Maple 计算并将结果返回到 MATLAB 7.0 命令窗口。因此可以说 MATLAB 7.0 中的符号运算是它处理数字运算的自然扩展。

进行符号计算，第一，计算以推理解析的方式进行，因此不受计算误差累积所带来的困扰；第二，符号计算可以给出完全正确的封闭解或任意精度的数值解（当封闭解不存在时）；第三，符号计算指令的调用比较简单，与经典教科书公式相近；第四，计算所需要的时间较长。

在 MATLAB 7.0 中，符号计算虽以数值计算的补充身份出现，但设计符号计算的相关指令、符号计算结果的图形化显示、符号计算程序的编写以及在线帮助系统都是十分完整和便捷的。

MATLAB 7.0 的升级和符号计算内核 Maple 的升级，决定了符号计算工具包的升级。但从用户使用的角度来看，这些升级所带来的变化相当细微。本章在相关的部分也给出了声明。

8.1　符号运算入门

科学与工程技术中的数值运算固然重要，但自然科学理论分析中各种各样的公式、关系式及其推导就是符号运算要解决的问题。它与数值运算一样，都是科学计算研究的重要内容。MATLAB 7.0 数值运算的对象是数值，而 MATLAB 7.0 符号运算的对象则是非数值的符号对象。所谓符号对象就是指代表非数值的符号字符串。下面列举一些实例来具体说明 MATLAB 7.0 的符号运算功能。

8.1.1　求解一元二次方程 $x^2+2x+2=0$ 的根

这是一个求解一元二次方程数值根的数字计算问题。根据求根公式，很容易得到方程的两个数值根是 $x_{1,2}=-1\pm j$。

而对于一元二次方程一般式 $ax^2+bx+c=0$ 来说，要求它的根，就必须根据一元二次方程求根公式，得到方程的根为 $x_{1,2}=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$。这就是一个求一元二次方程根的符号运算问题。

MATLAB 7.0 里的符号运算功能正是用来解决这类问题的。只要运行以下的函数命令：

```
solve('a*x^2+b*x+c=0')
```

即可求得一元二次方程的两个根：

```
ans =
 1/2/a*(-b+(b^2-4*a*c)^(1/2))
 1/2/a*(-b-(b^2-4*a*c)^(1/2))
```

求解一元二次方程数值根的计算问题也可以运行相同的 MATLAB 7.0 函数命令。例如求方程 $x^2+2x+2=0$ 的根时，可运行如下的语句：

```
solve('x^2+2*x+2=0')
```

即可得到方程的根：

```
ans =
 -1+i
 -1-i
```

8.1.2 求导数 $\frac{d}{dx}(\cos^2 x)$

根据复合函数求导公式，有 $\frac{d}{dx}(\cos^2 x)=2\cos x(-\sin x)=-2\sin x\cos x$。可以运行以下 MATLAB 7.0 语句求复合函数的导数：

```
x=sym('x');
diff(cos(x)^2)
```

即可求得导数：

```
ans =
-2*cos(x)*sin(x)
```

8.1.3 计算定积分 $\int_a^b x^2 dx$

计算定积分 $\int_a^b x^2 dx$ 即做运算 $\int_a^b x^2 dx=\frac{1}{3}x^3\Big|_a^b=\frac{1}{3}(b^3-a^3)$。可以运行以下两条 MATLAB 7.0 语句：

```
syms x a b;
int(x^2,a,b)
```

即得到定积分的结果：

```
ans =
1/3*b^3-1/3*a^3
```

8.1.4　求解一阶微分方程 $\frac{dy}{dt}=ay$

即作运算 $\int\frac{dy}{y}=\int adt$，$\ln y=at+c$，$y(t)=e^{at+c}=c_1e^{at}$。

可以运行以下 MATLAB 7.0 语句：

```
syms a y;
dsolve('Dy=a*y')
```

即求得一阶微分方程的解：

```
ans =
C1*exp(a*t)
```

从以上几个例子可以看出 MATLAB 7.0 符号运算的对象全是文字符号，算得的结果也是文字符号，同时符号运算基本上覆盖了初等数学以及高等数学中的绝大多数内容，而且这些运算都能够用 MATLAB 7.0 函数命令行来实现。

8.2　符号对象的创建和使用

符号数学工具箱中定义了一种新的数据类型，叫 sym 类。sym 类的实例就是符号对象，符号对象是一种数据结构，是用来存储代表符号的字符串的。在符号数学工具箱中，用符号对象来表示符号变量、符号表达式和符号矩阵。

本节将介绍如何借助 MATLAB 7.0 的符号数学工具箱来创建和使用符号变量、符号表达式和符号矩阵，同时介绍 MATLAB 7.0 的默认符号变量及其设置方法。

8.2.1　创建符号对象和表达式

在一个 MATLAB 7.0 程序中，作为符号对象的符号常量、符号变量、符号函数以及符号表达式，可以使用函数命令 sym()、syms()加以规定和创建，利用 class()可以测试建立的操作对象为何种操作对象类型以及是否为符号对象类型。

（1）函数命令 sym()

函数命令 sym()的调用格式有如下两种：

```
S＝sym(A,flag)
S＝sym('A',flag)
```

命令公式是由 *A* 来建立一个符号对象 *S*，其类型为 sym 类型。如果 *A*（不带单引号）是一个数字、数值矩阵或数值表达式，则输出是将数值对象转换成的符号对象。如果 *A*（带单引号）是一个字符串，输出则是将字符串转换成的符号对象。

其中 *flag* 为转换的符号对象应该符合的格式。如果被转换的对象为数值对象，*flag* 可以有如下选择：

- 'd'—最接近的十进制浮点精确表示；

- 'e'—带（数值计算时）估计误差的有理表示；
- 'f'—十六进制浮点表示；
- 'r'—为缺省设置时，最接近有理表示的形式。

对于被转换对象为字符串时，*flag* 有如下几种选项：

- 'positive'—限定 *A* 为正的实型符号变量；
- 'real'—限定 *A* 为实型符号变量；
- 'unreal'—限定 *A* 为非实型符号变量。

（2）函数命令 syms()

函数命令 syms()的调用格式为：

```
syms s1 s2 s3 flag;
```

该命令可以建立 3 个或多个符号对象，如 *s*1、*s*2、*s*3。同样，*flag* 为对转换格式的限定，具体选项同上。

（3）函数命令 class()

函数命令 class()的调用格式如下：

```
str＝class(object)
```

class()命令将返回对象的数据类型。

以下就分别来介绍在不同情况下这些函数的具体使用方法。

1．符号常量

符号常量是一种符号对象。数值常量如果作为函数命令 sym()的输入参量，这就建立了一个符号对象——符号常量，虽然看上去它是一个数值量，但它已是一个符号对象了。创建这个符号对象可以用 class()函数来检测其数据类型。

【例 8.1】对数值 3/4 创建符号对象并检测数据的类型。

【解】可以用以下 MATLAB 7.0 语句来创建符号对象并检测数据的类型：

```
a = 3/4;
b = '3/4';
c = sym(3/4);
d = sym('3/4');
classa = class(a)
classb = class(b)
classc = class(c)
classd = class(d)
```

语句的执行结果为：

```
classa = double
classb = char
classc = sym
classd = sym
```

即 *a* 实双精度浮点数值类型；*b* 实字符类型；*c* 和 *d* 都是符号对象类型。

2．符号变量

变量是程序语言的基本元素之一。MATLAB 7.0 符号运算中，符号变量是内容可变的符号对象。符号变量通常是指一个或几个特定的字符，不是指符号表达式，甚至可以将一个符号表达式赋值给一个符号变量。符号变量有时也称为自由变量。符号变量与 MATLAB 7.0 数值变量名称的命名规则相同：

- 变量名可以由英文字母、数字和下划线组成；
- 变量名应以英语字母开头；
- 组成变量名的字母长度不大于 31 个；
- MATLAB 7.0 中区分大小写英语字母。

在 MATLAB 7.0 中可以用函数命令 sym()和 syms()来建立符号变量。

【例 8.2】用函数命令 sym()和 syms()建立符号变量 a、b、c。

【解】

（1）用函数命令 sym()来创建符号对象并检测数据的类型

用 sym()命令来创建符号对象并检测数据类型的代码如下：

```
a = sym('a');
b = sym('b');
c = sym('c');
classa = class(a)
classb = class(b)
classc = class(c)
```

语句执行的结果如下：

```
classa = sym
classb = sym
classc = sym
```

可以看出得到的 3 个变量均为符号对象。

（2）用函数命令 syms()来创建符号对象并检测数据的类型

用 syms()命令来创建符号对象并检测数据类型的代码如下：

```
syms a b c;
classa = class(a)
classb = class(b)
classc = class(c)
```

语句执行的结果如下：

```
classa = sym
classb = sym
classc = sym
```

从上面的两种实现方法可以看出，对于需要同时定义多个符号变量的情况，使用 sym()命令分别定义显得过于繁琐，而可以采用 syms()命令进行代替。MATLAB 7.0 提倡用 syms()命令，因为它需要很少的书写，比较符合 MATLAB 7.0 符号运算简洁的特点。

3．符号表达式、符号函数与符号方程

表达式也是程序设计语言的基本元素之一。在 MATLAB 7.0 符号运算中，符号表达式是由符号常量、符号变量、符号函数运算符以及专用函数连接起来的符号对象。符号表达式有两类：符号函数与符号方程。符号函数不带等号，而符号方程是带等号的。

在 MATLAB 7.0 中，同样采用命令 sym()或 syms()来建立符号表达式。下面看一个建立符号函数的例子。

【例 8.3】用函数命令 sym()与 syms()建立符号函数 *f*1、*f*2、*f*3。

【解】用函数命令 sym()与 syms()来创建符号函数：

```
syms x y z;
f1 = x*y/z;
f2 = x^2+y^2+z^2;
f3 = f1/f2;
```

【例 8.4】用函数命令 sym()来创建符号方程 *e*1、*e*2、*e*3。

【解】具体的实现命令如下：

```
e1 = sym('a*x^2+b*x+c')
e2 = sym('sin(x)^2+2*cos(x)=1')
e3 = sym('Dy-y=x')
```

4．符号矩阵

元素是符号对象（非数值符号的字符符号即符号变量与符号形式的数即符号常量）的矩阵叫做符号矩阵。符号矩阵既可以构成符号矩阵函数，也可以构成符号矩阵方程，它们都是符号表达式。

【例 8.5】用函数命令 sym()建立符号矩阵 *m1*、*m2*。

【解】具体的实现命令如下：

```
m1 = sym('[ab bc cd;de ef fg;h I j]');
m2 = sym('[1 12;23 34]');
```

从上面的例子可以看出，创建符号矩阵的方法可以概括为两种：sym()指令直接输入法，如 *m*1 的实现；数值矩阵转换法，如 *m*2。这里需要重点指出的是第二种方法，数值矩阵转换法，把这种方法同 MATLAB 7.0 种许多数值矩阵的生成函数配合起来，可以产生出许多特殊的符号矩阵。

8.2.2　符号对象的基本运算

由于新版 MATLAB 7.0 采用了重载技术，使得构称符号计算表达式的运算符和基本函数，无论在形状、名称上，还是在使用方法上，都与数值计算中的运算符和基本函数几乎完全相同。这无疑给用户带来了极大的方便。

下面就符号计算种的基本算符和函数作一些简要的归纳。

1．基本运算符

- 运算符“＋”、“－”、“＊”、“\”、“/”、“^”分别实现矩阵的加、减、乘、左除、右除和求幂运算。
- 运算符“．＊”、“．/”、“．\”、“．^”分别实现“元素对元素”的数组乘、左除、右除和求幂运算。
- 运算符“’”、“．’”分别实现矩阵的共轭转置和非共轭转置。

2．关系运算符

在符号对象的比较中，没有“大于”、“大于等于”、“小于”和“小于等于”的概念，而只有是否“等于”的概念。

运算符“＝＝”和“～＝”分别对运算符两边的对象进行“相等”、“不等”的比较。当事实为“真”时，返回结果 1；否则，返回结果 0。

3．三角函数、双曲函数以及它们的反函数

除 atan2 仅能用于数值计算外，其余的三角函数（如 sin）、双曲函数（如 cosh）及它们的反函数（如 asin、acosh），无论在数值计算还是在符号计算中，它们的使用方法都相同。

4．指数、对数函数

在数值、符号计算中，函数 sqrt、exp 和 expm 的使用方法完全相同。至于对数函数，在 MATLAB 6.X 版本中，符号计算中只有自然对数 log，而没有数值计算中的 log2、log10。而在 MATLAB 7.X 版本中，增加了 log2 和 log10 函数，其用法同数值运算中完全相同。

5．复数函数

复数函数涉及复数的共轭（conj）、实部（real）、虚部（imag）和模（abs）的函数，在符号、数值计算中的使用方法相同。但需要注意的是，在符号运算中，MATLAB 7.0 没有提供求相角的指令。

6．矩阵代数指令

在符号运算中，MATLAB 7.0 提供的常用矩阵代数指令有 diag、triu、tril、inv、det、rank、rref、null、colspace、poly、expm、eig 和 svd。它们的用法几乎与数值计算中的情况完全一样，只有 svd 稍微不同，后面将有详细的介绍。

8.3　任意精度数学计算

符号计算的一个非常显著的特点是由于计算过程中不会出现舍入误差，从而可以得到任意精度的数值解。如果希望计算结果精确，那么就可以牺牲计算时间和存储空间，用符号计算来获得足够高的计算精度。

在符号运算工具箱中有 3 种不同类型的算术运算：

- 数值类型：MATLAB 7.0 的浮点算术运算；
- 有理数类型：Maple 的精确符号计算；
- VPA 类型：Maple 的任意精度算术运算。

这 3 种运算各有利弊，在计算时应根据计算精度、时间、存储控件的要求进行合理的选择。首先来看一下浮点运算和有理数运算的特点。下面是两种运算的例子：

```
format long                    %指定输出格式
1/2+1/3                        %一般的浮点运算
sym(1/2+1/3)                   %有理数运算（符号运算）
```

由以上语句得到结果代码如下：

```
ans =
    0.83333333333333
ans =
5/6
```

浮点运算是最快的运算，需要的计算机内存最小，但是结果不精确。MATLAB 7.0 双精度数输出的数字位数由 format 命令控制，但它内部采用的是由计算机硬件提供的八位浮点的表示方法。而且，在上面的浮点运算中，存在三种舍入误差，第一是来自 1/3 的除法舍入误差；第二是来自将 1/2 加到 1/3 的结果上产生的舍入误差；第三是来自将二进制结果转化为十进制输出时的舍入误差。

而符号运算中的有理数算术运算，它所需要的时间和内存开销都是最大的。只要有足够的内存和足够长的计算时间，总能产生精确的结果。

一般符号计算的结果都是字符串，特别是一些符号计算结果从形式上看来是数值，但从变量类型上说，它们仍然是字符串，例如上面例子中的“5/6”实际上就是字符串，而非数值。要从精确解中获得任意精度的解，并改变默认精度，把任意精度符号解变成“真正的”数值解，就需要用到 MATLAB 7.0 提供的如下几个函数。

1．digits(d)

调用该函数后的近似解的精度变成 *d* 位有效数字。*d* 的默认值是 32 位。

另外，调用 digits 命令（没参数）可以得到当前采用的数值计算的精度。

2．vpa(A,d)

求符号解 *A* 的近似解，该近似解的有效位数由参数 *d* 来指定。如果不指定 *d*，则按照一个 digits(d)指令设置的有效位数输出。

3．double(A)

把符号矩阵或任意精度表示的矩阵 ***A*** 转换成为双精度矩阵。

> 对于 vpa 函数的输入既可以是符号对象，也可以是数值对象，而其输出一定为符号对象。

【例 8.6】指令使用演示。

下面的命令首先创建一个符号矩阵，然后分别将它转换成任意精度矩阵和双精度矩阵。

```
A=[1.100 2.300 3.500;4.900 5.400 6;9.100 7.890 4.230];
```

```
S=sym(A)
```

生成符号矩阵如下：

```
S =
[     11/10,     23/10,          7/2]
[     49/10,      27/5,            6]
[     91/10, 789/100, 423/100]
```

转换成有效位为 4 的任意精度矩阵如下：

```
digits(4);
vpa(S)
ans =
[ 1.100, 2.300, 3.500]
[ 4.900, 5.400,       6.]
[ 9.100, 7.890, 4.230]
```

转换成双精度型的矩阵如下：

```
double(S)
ans =
    1.10000000000000    2.30000000000000    3.50000000000000
    4.90000000000000    5.40000000000000    6.00000000000000
    9.10000000000000    7.89000000000000    4.23000000000000
```

8.4　符号表达式的化简和替换

从前面几节可以发现，符号计算所得的结果往往比较繁琐，非常不直观。为此，MATLAB 7.0 专门提供了对符号计算结果进行化简和替换的函数，诸如因式分解、同类项合并、符号表达式的展开、符号表达式的化简和通分等，它们都是表达式的恒等变换。

8.4.1　符号表达式的化简

MATLAB 7.0 符号工具箱中提供了函数 collect、expand、horner、factor、simplify 和 simple 来实现符号表达式的化简。下面将分别介绍这些函数。

1．函数 collect()

函数实现功能为将符号表达式中同类项合并，具体调用格式有两种：

- R=collect(S)：将表达式 *S* 中的相同次幂的项合并。其中 *S* 可以是一个表达式，也可以是一个符号矩阵。
- R=collect(S,v)：将表达式 *S* 中 *v* 的相同次幂的项进行合并。如果 *v* 没有指定，则缺省地将含有 *x* 的相同次幂的项进行合并。

下面是 collect 函数调用的一个例子：

【例 8.7】

```
syms x t                                 %定义基本变量
f = (x-1)*(x-2)*(x-3)                    %定义符号表达式
collect(f)                               %合并 f 中 x 的同类项
```

结果为：

```
ans =
-6+x^3-6*x^2+11*x
```

2．函数 expand()

函数实现功能为将表达式进行展开。它的调用格式为：

- R = expand(S)：它将表达式 *S* 中的各项进行展开，如果 *S* 包含函数，则利用恒等变形将它写成相应的和的形式。该函数多用于多项式，有时也用于三角函数、指数函数和对数函数。

下面是函数 expand 应用的两个例子：

【例 8.8】多项式的展开示例：

```
syms x y;                                %定义基本变量
f = (x+y)^3;                             %创建符号表达式
f1 = expand(f)                           %展开多项式 f
```

得到结果代码如下：

```
f1 =
x^3+3*x^2*y+3*x*y^2+y^3
```

【例 8.9】三角函数的展开示例：

```
h = cos(x-y)                             %创建符号表达式
expand(h)                                %展开三角函数
```

得到结果代码如下：

```
ans =
cos(x)*cos(y)+sin(x)*sin(y)
```

3．函数 horner()

函数实现将符号表达式转换成嵌套形式。它的调用格式为：

- R = horner(S)：其中 *S* 是符号多项式矩阵，函数 horner 将其中每个多项式转换成它们的嵌套形式。

下面是函数 horner 应用的一个例子。

【例 8.10】

示例代码设置如下：

```
syms x y                                 %定义基本变量
f = x^3-6*x^2+11*x-6                     %定义符号多项式
horner(f)                                %将 f 转换成嵌套形式
```

得到结果代码如下：

```
ans =
-6+(11+(-6+x)*x)*x
```

4．函数 factor()

函数实现将符号多项式进行因式分解。它的调用格式为：

- factor(X)：如果 X 是一个多项式或多项式矩阵，系数是有理数，那么该函数将把 X 表示成系数为有理数的低阶多项式相乘的形式；如果 X 不能分解成有理多项式乘积的形式，则返回 X 本身。

下面是函数 factor 应用的一些例子：

【例 8.11】

示例代码设置如下：

```
syms x y a b
factor(x^3-y^3)
```

得到结果代码如下：

```
ans =
(x-y)*(x^2+x*y+y^2)
```

以下的指令将对如 x^(n+1)的多项式进行因式分解：

```
n = 1:9
x = x(ones(size(n)));
p = x.^n+1;
[p;factor(p)].'
```

输出的矩阵中，每行的第一个元素为分解前的多项式，第二个元素为分解后的多项式。当 n 为偶数时，x^$n+1$ 无法分解成有理多项式，所以返回它们自身。具体代码如下：

```
ans =
[        x+1,                          x+1]
[      x^2+1,                        x^2+1]
[      x^3+1,                (x+1)*(x^2-x+1)]
[      x^4+1,                        x^4+1]
[      x^5+1,        (x+1)*(x^4-x^3+x^2-x+1)]
[      x^6+1,            (x^2+1)*(x^4-x^2+1)]
[      x^7+1,  (x+1)*(1-x+x^2-x^3+x^4-x^5+x^6)]
[      x^8+1,                        x^8+1]
[      x^9+1,      (x+1)*(x^2-x+1)*(x^6-x^3+1)]
```

此外，函数 factor 还可以类似算术中质因数分解那样对正整数进行最简分解。当输入中的某个元素超过了 16 位，则必须先将该矩阵用函数 sym 定义成符号矩阵才能进行分解。这时，factor 是对符号整数进行分解。事实上，具有对整数进行分解功能的是在特殊功能函数工具箱中的 factor，它与这里讨论的符号工具箱中的 factor 同名。

下面的程序将对符号整数进行与上面类似的形式进行分解：

【例 8.12】

示例代码设置如下：

```
one = '1';
for n = 1:11
```

```
    N(n,:) = sym(one(1,ones(1,n)));
end
[N factor(N)]
```

得到的结果代码如下：

```
ans =

[                         1,                              1]
[                        11,                           (11)]
[                       111,                      (3)*(37)]
[                      1111,                    (11)*(101)]
[                     11111,                    (41)*(271)]
[                    111111,        (3)*(7)*(11)*(13)*(37)]
[                   1111111,                 (239)*(4649)]
[                  11111111,         (11)*(73)*(101)*(137)]
[                 111111111,            (3)^2*(37)*(333667)]
[                1111111111,          (11)*(41)*(271)*(9091)]
[               11111111111,              (513239)*(21649)]
```

5．函数 simplify()

根据一定的规则对表达式进行简化，它的调用格式为：

- R= simplify(S)：该函数是一个强有力的具有普遍意义的工具。它应用于包含和式、方根、分数的乘方、指数函数、对数函数、三角函数、Bessel 函数以及超越函数等的表达式，并利用 Maple 化简规则对表达式进行简化。其中 *S* 可以是符号表达式矩阵。

下面是 simplify 函数应用的一个例子。

【例 8.13】

```
S =sym('[(x^2+5*x+6)/(x+2);sqrt(16)]');
simplify(S);
```

得到的结果代码如下：

```
ans =
 x+3
   4
```

6．函数 simple()

寻找一个符号表达式的最简形式。它的调用格式有：

- r = simple(S)：用几种不同的算术简化规则对符号表达式进行简化，返回使表达式 *S* 变得简短的形式。如果 *S* 使符号表达式矩阵，则返回使整个矩阵变成最短的形式，而不一定使每一项都最短；如果不给定输出参数 *r*，该函数将显示所有使表达式 *S* 变短的简化形式，并返回其中最短的一个。
- [r,how] = simple(S)：不显示简化的中间结果，只显示寻找到的最短形式以及找到该形式所用的简化方法。返回值中，*r* 是符号表达式，*how* 是一个描述简化方法的字符串。

函数 simple 的目标是使表达式用最少的字符来表示。虽然并非表达式中的字符越少，表达式就越简洁，但采用这个标准往往能够得到满意的结果。为了达到这个最少字符的标准，simple 函数综合使用了以下几个函数进行不同方式的化简：

- 用函数 simplify 对表达式进行化简；
- 用函数 radsimp 对包含根式的表达式进行化简；
- 用函数 combine 把表达式中以求和形式、乘积形式、幂形式出现的各项进行合并；
- 用函数 collect 合并同类项；
- 用函数 factor 实现因式分解；
- 用函数 convert 把一种形式转换成另一种形式。

函数 simple 比较这些函数的结果，最终把最少字符作为标准。下面是函数 simple 应用的一些例子，读者可以从中体会到该函数在进行化简时的强大功能。

【例 8.14】

示例代码设置如下：

```
syms x
simple(cos(x)^2+sin(x)^2)
```

输出的结果代码如下：

```
simplify:                1
radsimp:                 cos(x)^2+sin(x)^2
combine(trig):           1
factor:                  cos(x)^2+sin(x)^2
expand:                  cos(x)^2+sin(x)^2
combine:                 1
convert(exp):     (1/2*exp(i*x)+1/2/exp(i*x))^2-1/4*(exp(i*x)-1/exp(i*x))^2
convert(sincos):         cos(x)^2+sin(x)^2
convert(tan):     (1-tan(1/2*x)^2)^2/(1+tan(1/2*x)^2)^2+4*tan(1/2*x)^2/(1+tan(1/2*x)^2)^2
collect(x):              cos(x)^2+sin(x)^2
mwcos2sin:               1
ans =                    1
```

由于函数 simple 采用多种方法进行化简，所以它往往能够改善函数 simplify 给出的结果，特别值得一提的是，simple 函数对于含有三角函数的表达式很有效。simple 调用的很多函数都能把三角多项式进行化简。下面是用 simple 化简三角多项式的示例代码：

```
simple(cos(3*acos(x)))
```

得到的结果代码如下：

```
ans =
4*x^3-3*x
```

8.4.2　符号表达式的替换

在 MATLAB 7.0 中，可以通过符号替换来使表达式的输出形式简化，从而可以得到比较简单的表达式。符号运算工具箱中提供了两个函数 subexpr 和 subs，用于实现符号对象的

替换。

1．函数 subexpr

函数 subexpr 将表达式中重复出现的字符串用变量代替，它的调用格式如下：

- [Y,SIGMA] = subexpr(S,SIGMA)：指定用变量 *SIGMA* 的值（必须为符号对象）来代替符号表达式（可以是矩阵）中重复出现的字符串。替换后的结果由 *Y* 返回，被替换的字符串由 *SIGMA* 返回；
- [Y,SIGMA] = subexpr(S,'SIGMA')：这种形式和上一种形式的不同在于第二个输入参数是字符或字符串，它用来替换符号表达式中重复出现的字符串。其他参数同上面的形式相同。

下面是函数 subexpr 的一个例子：

【例 8.15】

```
syms a x;
s = solve('x^3+a*x+1')
```

得到的结果代码如下：

```
s=
1/6*(-108+12*(12*a^3+81)^(1/2))^(1/3)-2*a/(-108+12*(12*a^3+81)^(1/2))^(1/3)

-1/12*(-108+12*(12*a^3+81)^(1/2))^(1/3)+a/(-108+12*(12*a^3+81)^(1/2))^(1/3)+1/2*i*3^(1/2)*(1/6*(-108+12*(1
2*a^3+81)^(1/2))^(1/3)+2*a/(-108+12*(12*a^3+81)^(1/2))^(1/3))

-1/12*(-108+12*(12*a^3+81)^(1/2))^(1/3)+a/(-108+12*(12*a^3+81)^(1/2))^(1/3)-1/2*i*3^(1/2)*(1/6*(-108+12*(1
2*a^3+81)^(1/2))^(1/3)+2*a/(-108+12*(12*a^3+81)^(1/2))^(1/3))
```

所得到的结果非常繁琐，虽然用函数 simple 不能化简它，但是 subexpr 函数则可以更为简洁，代码如下所示：

```
r = subexpr(s)
```

得到的结果代码如下：

```
sigma =
-108+12*(12*a^3+81)^(1/2)
r =
                                    1/6*sigma^(1/3)-2*a/sigma^(1/3)
  -1/12*sigma^(1/3)+a/sigma^(1/3)+1/2*i*3^(1/2)*(1/6*sigma^(1/3)+2*a/sigma^(1/3))
  -1/12*sigma^(1/3)+a/sigma^(1/3)-1/2*i*3^(1/2)*(1/6*sigma^(1/3)+2*a/sigma^(1/3))
```

2．函数 subs

函数 subs 可以用指定符号替换符号表达式中的某一特定符号。它的调用格式有：

- R = subs(S)：用工作空间中的变量值替代符号表达式 *S* 中的所有符号变量。如果没有指定某符号变量的值，则返回值中该符号变量不被替换；
- R = subs(S,New)：用新符号变量 *New* 替代原来符号表达式 *S* 中的默认变量。确定默认变量的规则与函数 findsym 的规则相同；

- R = subs(S,Old,New)：用新符号变量 *New* 替代原来符号表达式 *S* 中的变量 *Old*。当 *New* 是数值形式的符号时，实际上用数值代替原来的符号来计算表达式的值，只是所得结果仍然是字符串形式。

下面是 subs 应用的一些例子。

【例 8.16】

示例代码如下：

```
subs(exp(a*t),'a',-magic(2))                                    %用矩阵来替换符号变量
```

得到的结果代码如下：

```
ans =
[      exp(-t), exp(-3*t)]
[ exp(-4*t), exp(-2*t)]
```

除了本节介绍的一些简化和替换函数之外，符号数学工具箱还提供了一个将符号表达式显示起来比较美观的函数 pretty。它在缺省情况下以每一行 79 个字符宽的格式表示符号表达式，如果表达式中有多次出现的字符串，它会用%*n*（*n* 是整数）来代替它们。这是从 Maple 中继承而来的。下面举一个关于 pretty 的例子。

【例 8.17】

示例代码如下：

```
syms a x
s = solve('x^3+a*x+1')                          %求解一个方程
pretty(s)                                       %使用 pretty 函数来显示结果
```

得到的结果代码如下：

```
[                       1/3         a                       ]
[                  1/6 %1      - 2 -----                     ]
[                                  1/3                      ]
[                                %1                          ]
[                                                            ]
[          1/3       a              1/2 /        1/3         a   \]
[- 1/12 %1      + ----- + 1/2 I 3      |1/6 %1      + 2 -----|]
[                   1/3                |                 1/3|]
[                 %1                   \                 %1   /]
[                                                            ]
[          1/3       a              1/2 /        1/3         a   \]
[- 1/12 %1      + ----- - 1/2 I 3      |1/6 %1      + 2 -----|]
[                   1/3                |                 1/3|]
[                 %1                   \                 %1   /]


                                  3        1/2
            %1 := -108 + 12 (12 a    + 81)
```

8.5 符号矩阵的计算

在进行符号运算时，很多方面在形式上同数值计算都是相同的，不必再去重新学习一套关于符号运算的新规则，这给 MATLAB 7.0 用户带来了极大的方便。这里要介绍的符号对象的矩阵运算在形式上与数值计算中的运算十分相似，读者很容易掌握。

8.5.1 基本代数运算

在 MATLAB 7.0 中，符号对象的代数运算和双精度运算从形式上看是一样的。由于 MATLAB 7.0 中采用了符号的重载，用于双精度数运算的运算符同样可以用于符号对象。

符号对象的加减法运算必须满足下列原则，如果两个对象都是符号矩阵，那么它们必须大小相等。当然，符号矩阵也可以和符号标量进行加减运算，运算按照数组运算法则进行。

【例 8.18】符号矩阵的加减运算，示例代码设置如下：

```
syms a b c d                              %定义基本的符号变量
A = sym('[a b;c d]');                     %定义符号矩阵
B = sym('[2*a 3*b;c+a d+8]');             %定义符号矩阵
A+B                                       %计算符号矩阵的加法
```

得到的结果代码如下：

```
ans =
[    3*a,    4*b]
[2*c+a, 2*d+8]
```

关于符号矩阵的乘法（包括幂）运算必须要求参与运算的矩阵符合矩阵相乘的规则，例如第一个矩阵的列数等于第二个矩阵的行数等。对于这些基本的操作，限于篇幅，这里就不再赘述了，读者可以按照数值计算的部分进行练习。

8.5.2 线性代数运算

符号对象的线性代数运算和双精度数的线性代数运算一样，读者可以参阅本书第 2 章的有关内容来了解有关线性代数运算的一些规则和注意事项。这里仅举一个例子进行说明。

在下面的例子中，先生成数值希尔伯特矩阵，然后再将它转换成符号矩阵，并对它进行各种线性代数运算。读者可以从中体会符号对象线性代数运算的特点。示例代码如下：

```
H = hilb(3)                               %生成三阶希尔伯特数值矩阵
H = sym(H)                                %将数值矩阵转换成为符号矩阵
inv(H)                                    %求符号矩阵的逆矩阵
det(H)                                    %求符号矩阵的行列式
```

得到的结果代码如下：

```
三阶希尔伯特数值矩阵为：
H =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
```

```
      0.3333     0.2500     0.2000
符号矩阵：
H =
[    1, 1/2, 1/3]
[ 1/2, 1/3, 1/4]
[ 1/3, 1/4, 1/5]
符号矩阵的逆矩阵：
ans =
[      9,   -36,     30]
[   -36,   192, -180]
[     30, -180,   180]
符号矩阵的行列式：
ans =
1/2160
```

在上面的运算中，矩阵求逆和求行列式都采用的是符号计算，得到的结果都是精确解。下面用指定精度来计算上面的各项，将符号计算和数值计算进行一下比较。设置代码如下：

```
V = vpa(H,16)                                        %用 16 位精度转换成数值矩阵
```

得到一定精度的矩阵：

```
V =
[                    1., .5000000000000000, .3333333333333333]
[ .5000000000000000, .3333333333333333, .2500000000000000]
[ .3333333333333333, .2500000000000000, .2000000000000000]
```

上面把矩阵的每个元素的值按照四舍五入法转换成为 16 位有效数字，这时再来求矩阵的逆，这些舍入误差就会被放大。按照数值计算理论，放大的倍数就是矩阵的条件数。希尔伯特矩阵是一个严重病态矩阵，它的条件数非常大（3 阶矩阵的条件数大约为 500）。所以，误差将变的很大，这可以通过下面的计算中看到。

```
inv(V)                                               %对带有误差的矩阵进行求逆
```

求逆的结果代码如下：

```
ans =
[9.0000000000000612, -36.00000000000032, 30.0000000000003]
[-36.00000000000032, 192.00000000000166, -180.000000000001548]
[30.0000000000003, -180.000000000001548, 180.00000000000144]
```

可见，矩阵中的最后几位均为误差。下面再来看行列式的值：

```
det(V)                                               %求带有误差的矩阵的行列式
```

行列式的值如下：

```
ans =
.46296296296295907407407407740730e-3
```

下面通过修改 H(1,1)的值，将 H 变成奇异矩阵，并以它为例介绍奇异矩阵的特性：

```
syms s;
H(1,1) = s                                         %定义基本符号变量并替换 H(1,1)
```

得到 H(1,1)的值如下：

```
H =
[    s, 1/2, 1/3]
[ 1/2, 1/3, 1/4]
[ 1/3, 1/4, 1/5]
```

再来求行列式，代码设置如下：

```
Z = det(H)
```

得到新的行列式的值如下：

```
Z =
1/240*s-1/270
```

再求出 Z＝0 的 s 值，将它赋予 *sol*。

```
sol = solve(Z)
```

得到 *sol* 的值如下：

```
sol =
8/9
```

将 *sol* 替换 *s*，得到奇异矩阵：

```
H = subs(H,s,sol)
```

求新的符号矩阵的逆矩阵：

```
inv(H)
```

得到结果代码如下：

```
??? Error using ==> sym.inv
Error, (in inverse) singular matrix
```

由于 *H* 为奇异矩阵，所以求 *H* 的逆时将会给出错误信息。必须指出的是，虽然矩阵 *H* 是奇异矩阵，当采用任意精度进行计算时，无论求 *H* 的行列式还是逆矩阵，得到的结果均不是 0。这就是数值计算和符号计算的根本区别。

8.5.3 特征值分解

在 MATLAB 7.0 中，分别采用下面的函数来求符号方阵的特征值和特征向量：

- E = eig(A)：求符号方阵 *A* 的符号特征值 E；
- [v,E] = eig(A)：返回方阵 *A* 的符号特征值 E 和相应的特征向量 v。

与它们对应的任意精度计算的指令是 E = eig(vpa(A))和[v,E] = eig(vpa(A))。对于上一节最后给出的矩阵 *H*，由于它是奇异矩阵，肯定有一个特征值为 0：

```
[v,E] = eig(H)                                %求矩阵 H 的特征值和特征向量
```

得到的特征值和特征向量分别如下：

```
v =
[                    1,    28/153+2/153*12589^(1/2),    28/153-2/153*12589^(1/2)]
[                   -4,                           1,                           1]
[                 10/3, 292/255-1/255*12589^(1/2), 292/255+1/255*12589^(1/2)]
E =
```

```
[                    0,                    0,                    0]
[                    0, 32/45+1/180*12589^(1/2),                 0]
[                    0,                    0, 32/45-1/180*12589^(1/2)]
```

生成的矩阵 v 和 E 中，v 的每一列就是 H 的一个特征向量，相应的 E 的对角线元素就是 H 的特征值。

8.5.4　约当标准型

对矩阵进行相似变换时，会产生约当标准型（Jordan Canonical Form）。对于矩阵 *A*，求它的约当标准型，也就是找一个非奇异矩阵 *V*，使 J = V/A*V 最接近对角矩阵，其中的 *V* 称为转换矩阵。

当矩阵对称且所有特征值均互异时，其约当标准型就是其特征值所组成的对角矩阵，相应的转换矩阵由特征向量按列组成。对于非对称或有重特征值的情形，约当标准型的对角元素由特征值组成，但对角线上面有非零元素。

MATLAB 7.0 提供了函数 jordan 来求矩阵的约当标准形，它的调用格式有：

- J = jordan(A)：计算矩阵 *A* 的约当标准型。其中 *A* 可以是数值矩阵或符号矩阵；
- [V,J] = jordan(A)：除了计算矩阵 *A* 的约旦标准型 *J* 外，还返回相应的变换矩阵 *V*。

需要特别注意的是，函数 jordan 对矩阵元素值的极微小变化均特别敏感，这使得采用数值方法计算约当标准型非常困难，矩阵 *A* 的值必须精确地知道它的元素是整数或有理式（例如，没有舍入误差）。对于任意精度矩阵是不提倡求其约当标准型的。例如下面的示例代码：

```
A = sym([1 -3 -2;-1 1 -1;2 4 5]);                    %定义矩阵
[V,J] = jordan(A)                                    %求约当标准型
```

得到的结果代码如下：

```
V =
[ -1, -1,    1]
[    0, -1,    0]
[    1,    2,    0]
J =
[ 3, 0, 0]
[ 0, 2, 1]
[ 0, 0, 2]
```

8.5.5　奇异值分解

在符号数学工具箱中只有任意精度矩阵的奇异值分解才是可行的，其中一个重要原因就在于由符号计算产生的公式一般都太长、太复杂，而且没有太多的用处。用于对符号矩阵 *A* 进行奇异值分解的函数式 svd，它的调用格式如下：

- S = svd(A)：给出符号矩阵奇异值对角矩阵，其计算精度由函数 *digits* 来指定；
- [U,S,V] = svd(A)：输出参数 *U* 和 *V* 是两个正交矩阵，它们满足关系式：A = U*S*V'。

下面考虑一个矩阵 A，它的元素同时为 $\frac{1}{(i-j+1/2)}$，其中 i 和 j 分别为矩阵的行号和列号，示例代码设置如下：

```
for i = 1:5                                          %用 for 循环生成 A 矩阵
    for j = 1:5
        A(i,j) = 1/(i-j+1/2);
    end
end
```

得到 A 矩阵为：

```
A =
[    2,    -2, -2/3, -2/5, -2/7]
[  2/3,     2,   -2, -2/3, -2/5]
[  2/5,   2/3,    2,   -2, -2/3]
[  2/7,   2/5,  2/3,    2,   -2]
[  2/9,   2/7,  2/5,  2/3,    2]
```

鉴于 A 中的每个元素都是有理形式，可以采用指定精度计算得到较精确的解。命令如下：

```
digits(30)                                   %指定输出精度
S = svd(vpa(A))                              %求输入矩阵 A 的奇异值
```

得到的结果代码如下：

```
S =
 1.46186449332488823827862698271
 2.92186083491368812533417027502
 3.12952059354525758508947902807
 3.14127222045650822959569061467
 3.14158923834132037283159614880
```

8.6　符号微积分

微积分运算在数学计算中的重要性是不言自明的，整个高等数学就是建立在微积分运算的基础上的。同时微积分运算也是后面解符号微分方程的必要知识准备。

在符号数学工具箱中提供了一些常用的函数来支持具有重要基础意义的微积分运算，涉及的方面主要包括微分、求极限、积分、级数求和和泰勒级数等。下面来具体介绍符号运算在微积分中的使用方法。

8.6.1　符号表达式的微分运算

1．diff 函数

当创建了符号表达式后，就可以利用函数 diff 对它们进行微分运算。函数 diff 的调用格式有 3 种，它们的形式和作用分别如下：

- diff(S,'v')：将符号“v”视作变量，对符号表达式或符号矩阵 S 求取微分；
- diff(S,n)：将 S 中的默认变量进行 n 阶微分运算，其中默认变量可以用 findsym 函数确定，参数 n 必须是正整数；
- diff(S,'v',n)：将符号“v”视作变量，对符号表达式或矩阵 S 进行 n 阶微分运算。

例如，我们首先建立一个符号表达式，然后取相应的微分，示例代码设置如下：

```
syms a x                    %定义基本变量
f = sin(a*x)                %定义符号表达式
df = diff(f)                %对缺省变量 x 求微分
```

得到的结果代码如下：

```
df = cos(a*x)*a
```

再对符号表达式中的指定变量 a 求二阶微分：

```
dfa = diff(f,a,2)
```

得到的结果代码如下：

```
dfa = -sin(a*x)*x^2
```

2．jacobian 函数

微分运算也可以对列向量进行，所得的结果也是一个列向量。现在来考察笛卡尔坐标（x，y，z）和球坐标（r，θ，ϕ）的转换问题。坐标之间的转换关系为 $x = r\times\cos\theta\cos\phi$，$y = r\times\cos\theta\sin\phi$和 $z = r\times\sin\theta$。其中 θ对应于仰角或纬度，而ϕ代表方位角或经度。在数学中，它们之间转换的矩阵为雅可比矩阵。用函数 jacobian 可以计算这个转换矩阵 J，它的数学表达式为 $J = \dfrac{\partial(x,y,z)}{\partial(r,\theta,\phi)}$。可见求变换矩阵的本质还是求取微分。

函数 jacobian 的调用格式如下：

- R = jacobian(w,v)：其中 w 是一个符号列向量，v 是指定进行变换的变量所组成的行向量。

笛卡尔坐标和球坐标之间转换雅可比矩阵 J 可以用如下的命令求得：

```
syms r l f                                  %定义基本变量
x = r*cos(l)*cos(f);
y = r*cos(l)*sin(f);                        %建立转换关系
z = r*sin(l)
J = jacobian([x;y;z],[r l f])               %计算转换的雅可比矩阵
```

得到的结果代码如下：

```
J =
```

```
[       cos(l)*cos(f), -r*sin(l)*cos(f), -r*cos(l)*sin(f)]
[       cos(l)*sin(f), -r*sin(l)*sin(f),    r*cos(l)*cos(f)]
[                sin(l),              r*cos(l),                          0]
```

再来计算雅可比矩阵的行列式，并进行简化：

```
detJ = simple(det(J))
```

得到的结果代码如下：

```
detJ =
-cos(l)*r^2
```

> 注意：雅可比函数的第一个参数必须是列向量，第二个参数必须是行向量。此外，由于雅可比矩阵的行列式值一般是比较复杂的，我们可以对这些复杂的结果进行三角函数变换和简化，以便得到具有较为简便的形式。

8.6.2 符号表达式的极限

求微分的基本思想是当自变量趋近某个值时，求函数值的变化。“无穷逼近”是微积分的一个基本思想，求极限是非常普遍的。事实上，导数就是由极限给出的：

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

在 MATLAB 7.0 中，用函数 limit 来求表达式的极限。函数 limit 的调用格式如下：

- limit(F,x,a)：求当 x→a 时，符号表达式 F 的极限；
- limit(F,a)：符号表达式 F 采用默认自变量（可由函数 findsym 求得），该函数求 F 的自变量趋近于 a 时的极限值；
- limit(F)：符号表达式 F 采用默认自变量，并以 $a = 0$ 作为自变量的趋近值，从而求符号表达式 F 的极限值；
- limit(F,x,a,'right')或 limit(F,x,a,'left')：分别求取符号表达式 F 的左极限和右极限，即自变量从左边或右边趋近于 a 时的函数极限值。

例如，下面的命令分别计算 sin 函数的导数和幂函数的极限（指数函数）：

```
syms h n x                                    %定义基本变量
dc = limit((sin(x+h)-sin(x))/h,h,0)           %按照导数的定义求 sin 的导数
```

求得的结果代码如下：

```
dc =
cos(x)
```

求极限可以从两边趋近，从不同的方向趋近可能得到的结果也不相同。limit 函数第 4 种调用方式提供了求单边极限的功能。比如，下面的 3 个表达式的值时不同的。需要注意的是，MATLAB 7.0 对于没有定义的极限将返回 NaN：

$$\lim_{x \to 0} \frac{1}{x}, \lim_{x \to 0^-} \frac{1}{x}, \lim_{x \to 0^+} \frac{1}{x}$$

用下面的命令来验证这一结论：

```
limit(1/x,x,0)                          %求第一个表达式的值
```

```
limit(1/x,x,0,'left')                    %求第二个表达式的值
limit(1/x,x,0,'right')                   %求第三个表达式的值
```

得到的结果代码如下：

```
ans =
NaN                                         %没有意义，返回 NaN
ans =
-Inf                                        %结果为负无穷
ans =
Inf                                         %结果为正无穷
```

8.6.3　符号表达式的积分

数学中，积分和微分是一对互逆的运算。符号数学工具箱种提供了函数 int 来求符号表达式的积分，它的调用格式如下：

- R = int(S)：用缺省变量求符号表达式 S 的不定积分，缺省变量可用函数 findsym 确定；
- R = int(S,v)：用符号标量 v 作为变量求符号表达式 S 的不定积分值；
- R = int(S,a,b)：符号表达式采用缺省变量，该函数求缺省变量从 a 变到 b 时符号表达式 S 的定积分值。如果 S 是符号矩阵，那么积分将对各个元素进行分别进行，而且每个元素的变量也可以独立地由函数 findsym 来确定，a 和 b 可以是符号或数值标量；
- R = int(S,v,a,b)：符号表达式采用符号标量 v 作为标量，求当 v 从 a 变到 b 时，符号表达式 S 的定积分值。其他参数和上一种调用方式相同。

例如，下面用各种调用方式求符号表达式和符号矩阵的积分值，首先是用缺省变量对符号表达式求积分。

```
int(-2*x/(1+x^2)^2)                       %对符号表达式进行积分
```

得到的结果代码如下：

```
ans =
1/(1+x^2)
```

和微分运算比起来，积分的运算困难要大得多。当函数和积分不存在时，MATLAB 7.0 将简单地返回原来的积分表达式。例如，下面的命令试图求贝赛尔函数的积分：

```
a = int(besselj(l,z),0,1)                 %求贝赛尔函数的积分
```

得到的结果代码如下：

```
a =
-besselj(0,1)+1                           %积分失败，返回原表达式
```

在符号数学工具箱中，常数是积分时一个比较敏感的问题。例如表达式 $e^{-(kx)^2}$，当 k 为实数时，该表达式的值永远为正，并且当 x 的值趋于正负无穷时，该表达式的值趋于 0。但是 Maple 的内核并不把 k^2 或 x^2 看作正数，而只是简单地假定它们为符号，其值不定，没有丝毫的数学属性。用下面的命令计算该积分，将给出错误信息：

```
syms x k;                                 %定义积分变量
f = exp(-(k*x)^2)                         %定义积分函数
```

```
int(f,x,-inf,inf)                               %积分该函数
```

得到的结果代码如下：

```
ans =
PIECEWISE([csgn(k)/k*pi^(1/2), csgn(k^2) = 1],[Inf, otherwise])
```

从上面的结果我们发现，当 *k*^2 **大于** 0 **的情况下，积分可以得到，而当其他情况下积分为无穷大。因此，在计算前，应当首先定义** *k* **为实变量，这样才能得到正确的结果：**

```
syms k real                              %定义 k 为实变量
int(f,x,-inf,inf)                        %重新积分 f 函数
```

得到的结果代码如下：

```
ans =
signum(k)/k*pi^(1/2)
```

> **注意：上面的** *k* **既是** MATLAB 7.0 **工作控件中的变量，同时也是** Maple **内核工作空间中的一个实变量，命令"** clear k **"只能从** MATLAB 7.0 **的工作空间中删除符号对象** *k*，**但在** Maple **内核空间中还存在实变量** *k*。**可以利用"** syms k unreal **"命令从** Maple **内核空间中删除** *k*。

8.6.4 级数的求和

函数 symsum **用于对符号表达式进行求和。该函数的调用格式如下：**

- r = symsum(s,a,b)**：求符号表达式** *s* **中默认变量从** *a* **变到** *b* **时的有限和；**
- r = symsum(s,v,a,b)**：求符号表达式** *s* **中变量** *v* **从** *a* **变到** *b* **时的有限和。**

具体的示例代码如下：

```
syms x k                                 %定义基本变量
s1 = symsum(1/k^2,1,inf)                 %求无穷级数的和
s1 = symsum(x^k,k,0,inf)                 %求无穷级数的和
```

得到的结果代码如下：

```
s1 =
1/6*pi^2
s1 =
-1/(x-1)
```

8.6.5 泰勒级数

函数 taylor **用来求符号表达式的泰勒级数展开式，该函数的调用格式如下：**

- r = taylor(f)**：***f* **是符号表达式，其变量采用默认变量，该函数将返回** *f* **在变量等于** 0 **处作** 5 **阶泰勒展开时的展开式；**
- r = taylor(f,n,v)**：符号表达式** *f* **以符号标量** *v* **作为自变量，返回** *f* **的** *n*-1 **阶麦克劳林级数（即在** *v* = 0 **处作泰勒展开）展开式；**
- r = taylor(f,n,v,a)**：返回符号表达式** *f* **在** *v* = *a* **处作** *n*-1 **阶泰勒展开的展开式。**

例如，下面的命令将返回函数的 8 **阶泰勒展开式：**

```
syms x                          %定义基本符号变量
f = 1/(2+cos(x))                %定义 f 函数
r = taylor(f,8)                 %以 x 为自变量求 f 的泰勒展开式
```

得到的结果代码如下：

```
r =
1/3+1/18*x^2+1/216*x^4+1/6480*x^6
```

下面的命令将给出函数在 x = 2 处展开的前 3 个非零项：

```
g = exp(x*sin(x))               %定义函数
t = taylor(g,3,2)               %求 x = 2 处的泰勒展开式
```

得到的结果代码如下：

```
t =
exp(2*sin(2))+exp(2*sin(2))*(2*cos(2)+sin(2))*(x-2)+exp(2*sin(2))*(-sin(2)+cos(2)+2*cos(2)^2+2*cos(2)*s
in(2)+1/2*sin(2)^2)*(x-2)^2
```

下面的命令将求出函数 sin(x)*e-x 的麦克劳林级数：

```
ft = taylor(sym(sin(x)*exp(-x)),8)          %求函数的麦克劳林级数
```

得到的结果代码如下：

```
ft =
x-x^2+1/3*x^3-1/30*x^5+1/90*x^6-1/630*x^7
```

8.7　符号积分变换

在数学中，为了把较复杂的运算转化为比较简单的运算，经常采用一种变换手段。例如数量的乘积或商可以通过对数学变换成对数的和或差，然后再取反对数即可求得原来数量的乘积或商。这一变换方法的目的就是把比较复杂的乘除运算通过对数变换转化为简单的加减运算。

所谓积分变换，就是通过积分运算，把一类函数 A 变换成另一类函数 B，函数 B 一般是含有参量 α 的积分 $\int_a^b f(t)K(t,\alpha)dt$。这一变换的目的，就是把某函数类 A 中的函数 f(t)通过积分运算变成另一类函数 B 中的函数 F(α)。这里 K(t, α)是一个确定的二元函数，叫做积分变换的核。当选取不同的积分区间与变换核时，就成为不同的积分变换。f(t)叫做原函数，F(α)叫做象函数。在一定条件下，原函数与象函数两者一一对应，成为一个积分变换对。变换时可逆，有原函数求象函数叫做正变换，反之则是逆变换。

积分变换的理论与方法，在自然科学与工程技术的各个领域中都有着极广泛的应用，成为不可缺少的运算工具。所有变换的使用，都会极大地简化计算，有的变换则为开创新的学科奠定了基础。

以下要介绍 3 种积分变换，即 Fourier 变换、Laplace 变换与 Z 变换。

8.7.1 Fourier 变换

时域中的 f(t)与它在频域中的 Fourier 变换 F(ω)之间存在如下关系：

$$F(\omega)=\int_{-\infty}^{\infty}f(t)e^{-j\omega t}dt$$

$$f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(\omega)e^{jwt}d\omega$$

由计算机完成这种变换的途径有两条：一种是直接调用指令 fourier 和 ifourier 进行；另一种是根据上面的定义，利用积分指令 int 实现。下面只介绍 fourier 和 ifourier 的使用及相关注意事项。至于根据定义求变换，请读者自己完成。

- Fw = fourier(ft,t,w)：求时域函数 *ft* 的 Fourier 变换 *Fw*，*ft* 是以 *t* 为自变量的时域函数，*Fw* 是以圆频率 *w* 为自变量的频域函数；
- ft = ifourier(Fw,w,t)：求频域函数 *Fw* 的 Fourier 反变换，*ft* 是以 *t* 为自变量的时域函数，*Fw* 是以圆频率 *w* 为自变量的频域函数。

下面的示例代码将给出 Fourier 变换的具体实现：

```
syms t w                    %定义基本符号变量
ut = Heaviside(t);          %定义 0 时刻起跳的单位阶跃函数
UT = fourier(ut)            %进行 fourier 变换
```

得到的结果代码如下：

```
UT =
pi*dirac(w)-i/w
```

再来求 Fourier 反变换：

```
Ut = ifourier(UT,w,t)           %进行 fourier 反变换
```

得到的结果代码如下：

```
Ut =
heaviside(t)
```

同原先的时域函数相同。

> 注意：在 MATLAB 7.0 的符号数学工具箱中增加了 dirac 和 heaviside 两个函数，它们分别是单位脉冲函数和阶跃函数。在老版本的 MATLAB 中，如果需要调用这两个函数，则必须从 Maple 函数库中调用，因为原先的 MATLAB 本身没有这样的函数提供。

8.7.2 Laplace 变换

Laplace 变换和反变换的定义为：

$$F(s)=\int_{0}^{\infty}f(t)e^{-st}dt$$

$$f(t)=\frac{1}{2\pi j}\int_{c-j\infty}^{c+j\infty}F(s)e^{st}ds$$

与 Fourier 变换相似，Laplace 变换与反变换的实现也有两条途径。直接调用指令 laplace 和 ilaplace 进行，或者根据上面的定义，利用积分指令 int 实现。比较而言，直接使用 laplace 和 ilaplace 指令实现变换较为简洁。具体使用方法如下：

- Fs = laplace(ft,t,s)：求时域函数 *ft* 的 Laplace 变换 *Fs*，*ft* 是以 t 为自变量的时域函数，*Fs* 是以复频率 s 为自变量的频域函数；
- ft = ilaplace(Fs,s,t)：求频域函数 *Fs* 的 Laplace 反变换 *ft*，*ft* 是以 t 为自变量的时域函数，*Fs* 是以复频率 s 为自变量的频域函数。

下面的例子给出 Laplace 变换的具体实现：

求 $\begin{bmatrix}\delta(t-a) & u(t-b)\\ e^{-at}\sin bt & t^2\cos 3t\end{bmatrix}$ 的 Laplace 变换。

```
syms t s                                          %定义基本符号变量
syms a b positive                                 %对常数进行“限定性”设置
Mt = [dirac(t-a),heaviside(t-b);exp(-a*t)*sin(b*t),t^2*cos(3*t)];        %定义输入矩阵
MS = laplace(Mt,t,s)                              %进行 Laplace 变换
```

得到的结果代码如下：

```
MS =
[                    exp(-s*a),                  exp(-s*b)/s]
[      1/b/((s+a)^2/b^2+1), 2/(s^2+9)^3*(s^3-27*s)]
```

8.7.3　Z 变换

一个离散因果序列的 Z 变换及其反变换定义为：

$$F(z)=\sum_{n=0}^{\infty}f(n)z^{-n}$$

$$f(n)=Z^{-1}\{F(z)\}$$

涉及 Z 反变换具体计算的方法，最常见的有 3 种，分别是幂级数展开法、部分分式展开法和围线积分法。MATLAB 7.0 的符号数学工具箱中采用了围线积分法设计了求取 Z 反变换的 iztrans 指令，相应的数学表达式是 $f(n)=\frac{1}{2\pi j}\int_T F(z)z^{n-1}dz$。具体的命令格式如下：

- FZ = ztrans(fn,n,z)：求时域函数 *fn* 的 Z 变换 *FZ*，*fn* 是以 n 为自变量的时域序列，*FZ* 是以复频率 z 为自变量的频域函数；
- fn = iztrans(FZ,z,n)：求频域函数 *FZ* 的 Z 反变换 *fn*，*fn* 是以 n 为自变量的时域序列，*FZ* 是以复频率 z 为自变量的频域函数。

下面的例子给出 Laplace 变换的具体实现：

求序列 $f(n)=\begin{cases}0 & n<0\\ 2 & n=0\\ 6(1-0.5^n) & n>0\end{cases}$ **的 Z 变换，并用反变换验算。**

示例代码如下：

```
%首先来构造单位函数并对其性能进行验证
syms n
Delta = sym('charfcn[0](n)');          %定义单位函数δ(n)
D0 = subs(Delta,n,0);                  %计算δ(0)
D15 = subs(Delta,n,15);                %计算δ(15)
disp('[D0,D15]');
disp([D0,D15])
```

得到的结果代码如下：

```
[D0,D15]
[ 1, 0]
```

再来求序列 f(n)的 Z 变换，示例代码如下：

```
syms z
fn = 2*Delta+6*(1-(1/2)^n);
FZ = ztrans(fn,n,z);
disp('FZ = ');
pretty(FZ)
```

得到的结果代码如下：

```
FZ =
                      z          z
              2 + 6 ----- - 12 -------
                    z - 1      2 z – 1
```

最后 Z 反变换，示例代码如下：

```
FZ_n = iztrans(FZ,z,n)
```

得到的结果代码如下：

```
FZ_n =
2*charfcn[0](n)+6-6*(1/2)^n
```

> **说明：在输入函数的定义上，我们引入了一个函数** charfcn**，需要说明的是，这个函数是** Maple V5 **中的表达式的指征函数（**Characteristic function**），它满足** $charfcn[A](x)=\begin{cases}1 & x\in A\\ 0 & x\notin A\end{cases}$。

8.8　符号方程求解

从学习代数开始，我们就一直在探索关于方程的求解理论，从最初的代入消元法和加减消元法到数值计算中的牛顿迭代法、高斯消元法，一直到微分方程的求解理论，方程在数学中的重要性就包含在这么漫长的探索、深化过程中。MATLAB 7.0 为符号方程的求解提供了强有力的支持。

符号方程根据其中涉及到的运算类别，可以分为代数方程和微分方程。其中代数方程只涉及到符号对象的代数运算，相对比较简单，它还可以细分为线性方程和非线性方程两类。前者往往可以很容易地求得所有解，但是对于后者来说，却经常容易丢掉一些解，这时就必须借助绘制函数图形，通过图形来判断方程解的个数。

微分方程的求解稍微复杂一些，它按照自变量的个数，可以分为常微分方程和偏微分方程。偏微分方程的求解在数学上相当复杂，而且理论体系也繁杂，用机器求解往往不能找到通行的方法，所以这里介绍用 MATLAB 7.0 求解常微分方程。

8.8.1　代数方程的求解

这里所讲的一般代数方程包括线性、非线性和超越方程等，求解指令是 solve。当方程组不存在符号解时，若又无其他自由参数，则 solve 将给出数值解。该指令的使用格式包括以下几种：

- g = solve(eq)：其中 *eq* 可以是符号表达式或不带符号的字符串，该函数将求解方程 eq = 0，其自变量采用默认变量，可以通过函数 findsym 来确定；
- g = solve(eq,var)：求解方程 eq = 0，其自变量由参数 *var* 指定。其中 *eq* 和上一种调用方式相同。返回值 *g* 是由方程的所有解构成的列向量；
- g = solve(eq1,eq2,…,eqn)：求解由符号表达式或不带符号的字符串 *eq*1，*eq*2，…，*eq*n 组成的方程组。其中的自变量为整个方程组的默认变量，即将函数 findsym 作用于整个方程组时返回的变量；
- g = solve(eq1,eq2,…,eqn,var1,var2,…,varn)：求解由符号表达式或不带等号的字符串 *eq*1，*eq*2，…，*eq*n 组成的方程组。其自变量由输入参数 *var*1，*var*2，…，*var*n 指定。

对于上面的 4 种调用方式，输出的解有 3 种情况：

- 对于单个方程单个输出参数的情况，将返回由多个解构成的列向量；
- 对于有和方程数目相同的输出参数的情况，方程组的解将分别赋给每个输出参数，并按照字母表的顺序进行排列；
- 对于只有一个输出参数的方程组，方程组的解将以结构矩阵的形式赋给输出参数。

下面给出一个代数方程的求解过程：

求方程组 $\begin{cases} uy^2+vz+w=0 \\ y+z+w=0 \end{cases}$ 关于 y，z 的解。

```
S = solve('u*y^2+v*z+w = 0','y+z+w = 0','y','z')
disp('S.y'),disp(S.y),disp('S.z'),disp(S.z)
```

得到解如下：

```
S =
    y: [2x1 sym]
    z: [2x1 sym]
%显示 y 的解
S.y
 -1/2/u*(-2*u*w-v+(4*u*w*v+v^2-4*u*w)^(1/2))-w
 -1/2/u*(-2*u*w-v-(4*u*w*v+v^2-4*u*w)^(1/2))-w
%显示 z 的解
S.z
 1/2/u*(-2*u*w-v+(4*u*w*v+v^2-4*u*w)^(1/2))
 1/2/u*(-2*u*w-v-(4*u*w*v+v^2-4*u*w)^(1/2))
```

注意：需要注意的是如果没有指定变量，findsym 将把 w，y 依次作为自变量，最终的结果将与现在不同，读者可以自己尝试。

8.8.2 微分方程的求解

在前面已经讲过：从数值计算角度看，与初值问题求解相比，微分方程边值问题的求解显得复杂和困难。对于应用数学工具去求解实际问题的科研人员来说，此时，不妨通过符号计算指令进行求解尝试。因为，对于符号计算来说，不论是初值问题，还是边值问题，其求解微分方程的指令形式都相同，且相当简单。

当然，符号计算可能花费较多的计算机资源，可能得不到简单的解析解或封闭形式的解，甚至无法求解。既然没有万能的微分方程一般解法，那么：求解微分方程的符号法和数值法有很好的互补作用。

函数 dsolve 用来求常微分方程的符号解。在方程中，用大写字母 D 表示一次微分，D2、D3 分别表示二次、三次微分运算。以此类推，符号 D2y 表示 $\frac{d^2y}{dt^2}$。函数 dsolve 把 d 后面的字符当作因变量，并默认所有这些变量对符号 t 进行求导。函数 dsolve 的调用方式有：

- r = dsolve('eq1,eq2,…','cond1,cond2,…','v')：求由 $eq1$，$eq2$，…指定的常微分方程的符号解。常微分方程以变量 v 作为自变量，参数 $cond1$，$cond2$，…用于指定方程的边界条件或者初始条件。如果 v 不指定，将默认 t 为自变量；
- r = dsolve('eq1','eq2',…,'cond1','cond2',…,'v')：求由 $eq1$，$eq2$，…指定的常微分方程的符号解。这些常微分方程都以 v 作为自变量。这些单独输入的方程的最大允许个数为 12。其他参数与上一种调用方式相同。

微分方程的初始条件或边界条件都以变量 v 作为自变量，其形式为 $y(a)=b$ 或 $Dy(a)=b$，其中 y 是微分方程的因变量，a 和 b 是常数。如果指定的初始条件和边界条件

比方程中的因变量个数少，那么所得的解中将包含积分常数 $C1$、$C2$ 等。

函数 dsolve 的输出结果同 solve 类似，既可以用和因变量个数相同数目的输出参数分别接收每个变量的解，也可以把方程的解写入一个结构数组中。

下面举两个常微分方程求解的例子：

求 $\frac{dx}{dt}=y$，$\frac{dy}{dt}=-x$ 的解。示例代码如下：

```
S = dsolve('Dx = y,Dy = -x');
disp([blanks(12),'x',blanks(21),'y']),disp([S.x,S.y])
```

得到的结果代码如下：

```
            x                     y
[ -C1*cos(t)+C2*sin(t),    C1*sin(t)+C2*cos(t)]
```

求解两点边值问题：$xy''-2y'=x^2$，$y(1)=0$，$y(2)=5$。

```
y = dsolve('x*D2y-2*Dy = x^2','y(1) = 0,y(5) = 0','x')
```

得到的结果代码如下：

```
y =
1/3*x^3*log(x)-1/9*x^3+1/3*(1/3-125/124*log(5))*x^3+125/372*log(5)
```

注意：在本例程中如果用数值解法，则比较复杂，有兴趣的读者可以自己尝试。

8.9　可视化数学分析界面

MATLAB 7.0 的符号数学工具箱为符号函数可视化提供了一组简便易用的指令。本章着重介绍两个进行数学分析的可视化界面，即图示化符号函数计算器（由指令 funtool 引出）和泰勒级数逼近分析界面（由指令 taylortool 引出）。

8.9.1　图示化符号函数计算器

对于习惯使用计算器或者只是向作一些简单的符号运算与图形处理的读者，MATLAB 7.0 提供的图示化符号函数计算器是一个较好的选择。该计算器功能虽简单，但操作方便，可视性强。

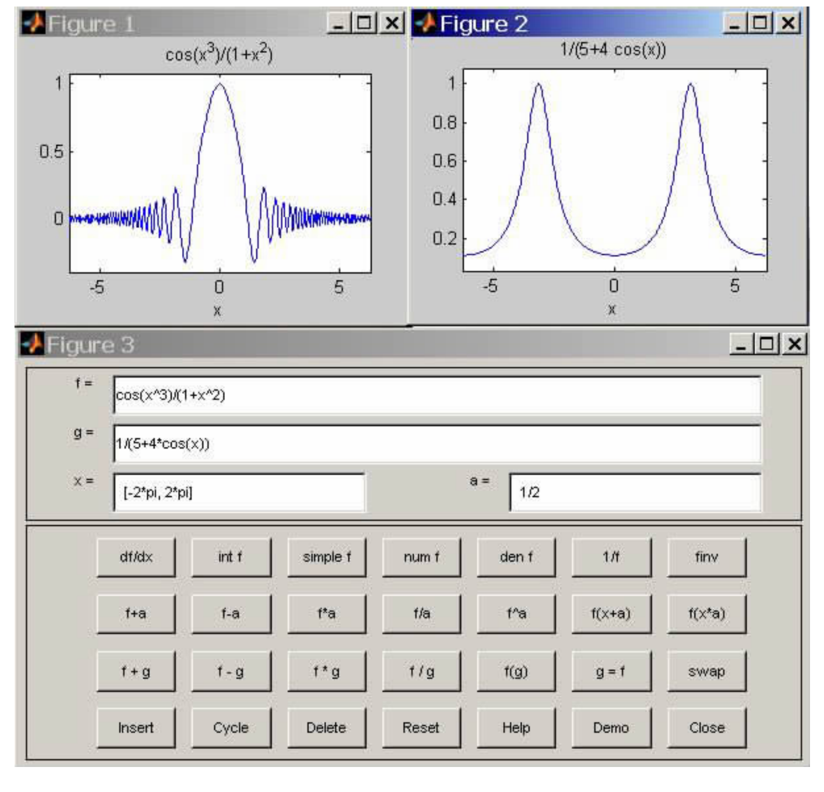


图 8-1　**图示化符号函数计算器界面**

在如图 8-1 所示的图示化符号函数计算器是由 2 个图形窗口（Figure No.1 和 Figure No.2）与一个函数运算控制窗口（Figure No.3）3 个独立窗口组成。在任何时候，两个图形窗口只有一个处于激活状态。函数运算控制窗口上的任何操作都只能对被激活的函数图形窗口起作用，即被激活的函数图像可随运算控制窗口的操作而作相应的变化。

（1）第 1 排按键只对 f 起作用，如求导、积分、简化、提取分子和分母、计算 $1/f$ 以及求反函数。

（2）第 2 排按键处理函数 f 和常数 a 之间的加、减、乘、除等运算。

（3）第 3 排的前 4 个按键对两个函数 f 和 g 之间进行算术运算。第 5 个按键求复合函数，第 6 个按键的功能是把 f 函数传递给 g，最后一个按键 swap 是实现 f 和 g 的互换。

（4）第 4 排按键用于对计算器自身进行操作。funtool 计算器有一张函数列表 fxlist。这 7 个按键的功能依次是：

- Insert：把当前激活窗的函数写入列表；
- Cycle：依次循环显示 fxlist 中的函数；
- Delete：从 fxlist 列表中删除激活窗的函数；
- Reset：使计算器恢复到初始调用状态；
- Help：获得关于界面的在线提示说明；
- Demo：自动演示。

8.9.2　泰勒级数逼近分析器

在 MATLAB 7.0 指令窗口中运行以下指令，将引出如图 8-2 所示的泰勒逼近分析界面。

```
taylortool
```

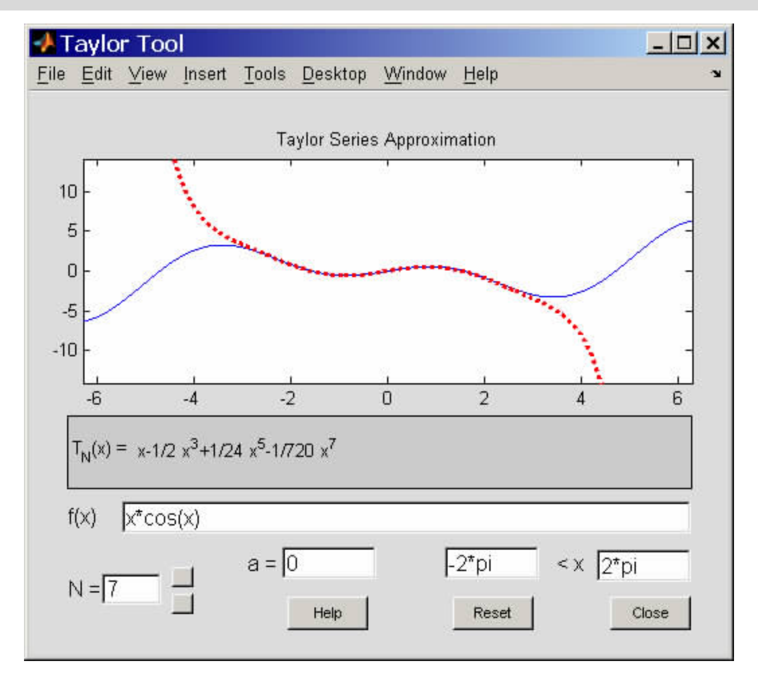


图 8-2　泰勒级数逼近分析界面

- 该界面用于观察函数 f(x)在给定区间上被 N 阶泰勒多项式 T_N（x）逼近的情况；
- 函数 f(x)的输入方式有两种，直接由指令 taylortool(fx)引入，或者在界面的 f(x)栏中直接键入表达式，然后回车输入；
- 界面中 N 被缺省地设置为 7，可以用其右侧的按键改变阶次，也可以直接写入阶次；
- 界面上的 *a* 是级数的展开点，缺省值为 0；
- 函数的观察区被缺省地设置为（-2π,2π）。

8.10　Maple 接口

相对于 Maple 中的 2000 多条符号计算指令而言，前面介绍的函数只是其中的一小部分。为了在 MATLAB 7.0 中进一步利用 Maple 的符号计算能力，要求用户必须对 Maple 有较深的理解。符号数学工具箱为此提供了两个函数，即 sym 和 maple。

sym 和 maple 这两个指令可以实现对 Maple 绝大多数符号计算指令的调用，利用它们可以大大扩充 MATLAB 7.0 的符号计算功能。下面将通过示例分别介绍两个指令的使用。

8.10.1　利用 sym 函数调用 Maple 函数

例如，通过函数 sym 来调用 Maple 的阶乘函数 k !，代码设置如下：

```
kfac = sym('k!');                                     %调用 Maple 函数
```

这相当于定义了一个可以用来计算正整数阶乘的符号函数。利用它来计算一下阶乘 7！和 n !，相应的命令如下：

```
syms k n                                              %定义基本变量
subs(kfac,k,7)                                        %计算阶乘
subs(kfac,k,n)                                        %计算阶乘
```

得到的结果代码如下：

```
ans =
5040
ans =
n!
```

8.10.2 利用 maple 函数调用 Maple 函数

采用 maple 函数可以直接调用 Maple 中的函数。这个函数采用符号对象、字符串和双精度数作为输入参数，返回和输入参数对应的符号对象、字符串和双精度数。也可以利用函数 maple 来测试用户自己编写的符号数学程序。

假定要写一个 M 文件，用来计算两个多项式和两个整数的最大公约数。为此，首先要知道怎样调用 Maple 中求取最大公约数的函数。可以使用命令 mhelp 来获得这方面的信息。输入命令“ mhelp gcd ”即可获得 Maple 中用于求取最大公约数的函数 gcd 的相关信息。

当已经知道了如何调用函数 gcd 后，就可以编写一个简单的 M 文件 gcd.m 来调用这个 Maple 函数，从而轻松地求得两个多项式或整数的最大公约数。M 文件的内容如下：

```
function g = gcd(a,b)
g = maple('gcd',a,b);
```

这样就可以简单地调用该文件，并利用 Maple 函数求最大公约数。示例代码如下：

```
syms x y                                              %定义基本变量
z = gcd(x^2-y^2,x^3-y^3)                              %求多项式的最大公约数
```

得到的结果代码如下：

```
z =
-y+x
```

可见，利用 maple 函数直接调用 Maple 中的函数，将大大地扩充 MATLAB 7.0 的符号计算功能。如果读者有什么特殊的符号计算要求，而不能由 MATLAB 7.0 来完成，不妨通过调用 Maple 函数来解决。

第 9 章　文件 I/O

MATLAB 系统具有直接对磁盘文件进行访问的功能，用户不仅可进行高层的程序设计，必要时还可进行低层次磁盘文件的读写操作，增强了 MATLAB 程序设计的灵活性。

MATLAB 内建有很多有关文件输入和输出的函数，使用者可以很方便地对二进制文件或 ASCII 文件进行打开、关闭和存储等操作。这些函数是基于 C 语言的文件 I/O 函数的，简单易学。

9.1　打开和关闭文件

9.1.1　打开文件

根据操作系统的要求，在程序要使用或者创建一个磁盘文件时，必须向操作系统发出打开文件的命令，使用完毕后，还必须通知操作系统关闭这些文件。

在 MATLAB 中，使用 C 语言中的同名函数 fopen 来完成这一功能，其语法为：

```
fid＝fopen('filename','permission')
```

其中 *filename* 是要打开的文件名称，*permission* 则表示要对文件进行处理的方式。可以是下列任一字符串：

- *'r'*：只读文件（*reading*）；
- *'w'*：只写文件，覆盖文件原有内容（如果文件名不存在，则生成新文件，*writing*）；
- *'a'*：增补文件，在文件尾增加数据（如果文件名不存在，则生成新文件，*appending*）；
- *'r+'*：读写文件（不生成文件，*reading and writing*）；
- *'w+'*：创建一个新文件或者删除已有文件内容，并可进行读写操作；
- *'a+'*：读取和增补文件（如果文件名不存在，则生成新文件）。

文件可以以二进制的形式或者文本形式打开（默认情况下是前者），在二进制形式下，字符串不会被特殊对待。如果要求以文本形式打开，则在 *permission* 字符串后面加*'t'*，例如*'rt+'*，*'wt+'*等。需要说明的是在 UNIX 下，文本形式和二进制形式没有什么区别。

fid 是一个非负整数，称为文件标识，对于文件的任何操作，都是通过这个标识值来传递的，MATLAB 通过这个值来标识已打开的文件，实现对文件的读、写和关闭等操作。正常情况下应该返回一个非负的整数，这个值是由操作系统设定的。如果返回的文件标识为'-1'，则表示 fopen 无法打开该文件，原因可能是该文件不存在，而以*'r'*或*'r+'*方式打开，也可能是用户无权限打开此文件。程序设计中，每次打开文件，都要进行打开操作是否正确的测定。如果要知道 fopen 操作失败的原因，可以使用下列方式调用：

```
[fid,message]=fopen('filename','r')
```

输出变量 *message* 中包含了文件打开操作结果的信息，示例代码设置如下：

```
[fid,message]=fopen('test.dat','r');
if fid==-1
    disp(message);
end
```

运行后在命令行显示为：

```
No such file or directory
```

如果是打开一个存在的文件，示例代码设置如下：

```
[fid,message]=fopen('sum2.m','r');
if fid==-1
    disp(message);
end
```

运行后分别查看 message 和 fid 的值如下：

```
>> message
message =
     ''
>> fid
fid =
     3
```

文件标识 fid＝1 和 fid＝2 是自动获取的，不必打开。如果要获取所有已打开的文件标识值，可用以下命令：

```
>> fids=fopen('all')
```

注意，查看错误信息时，也可用 MATLAB 的函数 ferror，其函数格式如下：

```
message=ferror(fid)
```

9.1.2 关闭文件

在进行完读写操作后，必须关闭文件，以免打开文件过多，造成系统资源浪费，命令为：

```
>> status=fclose(fid)
```

如果关闭成功，则返回 *status* 值为 0；否则返回－1。

上述命令是关闭了文件标识为 *fid* 的文件，如果要一次关闭所有打开的文件，则需执行下面的代码：

```
>> status=fclose('all')
```

用户可以通过检查 status 的值来确认文件是否关闭。

在某些情况下，可能需要用到暂存目录及临时文件，要取用系统的暂存目录，可用 tempdir 命令：

```
>> directory=tempdir
directory =
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\
```

要打开一个临时文件，可用 tempname 命令：

```
>> filename=tempname
filename =
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\tp145834
>> fid=fopen(filename,'w');
```

注意：打开和关闭文件的操作都比较费时，尽量不要将它们置于循环中，以提高程序执行的效率。

9.2　读取二进制文件

MATLAB 中函数 fread 可以从文件中读取二进制数据，将每一个字节看成一个整数，将结果写入一个矩阵返回。最基本的调用形式为：

```
a=fread(fid)
```

其中，*fid* 是从 fopen 中得来的文件标识。MATLAB 读取整个文件并将文件指针放在文件末尾处（在后面的 feof 命令介绍中有详细解释）。

例如，文件 *t_b.m* 内容如下：

```
a=[15,20,25,30,35];
b=[1554.88,1555.24,1555.76,1556.20,1556.68];
figure(1)
plot(a,b)
```

用 fread 函数读取此文件，操作如下：

```
>> fid=fopen('t_b.m','r');
>> data=fread(fid);
```

如要验证，可输入如下代码：

```
>> disp(char(data'))
a=[15,20,25,30,35];
b=[1554.88,1555.24,1555.76,1556.20,1556.68];
figure(1)
plot(a,b)
```

如果不用 char 将 *data* 转换为 ASCII 字符，则输出的是一组整数，取 *data* 的转置是为了方便阅读。

函数 fread 返回矩阵的大小和形式是可控的，通过 fread 的第 2 个输入变量来实现：

```
a=fread(fid,size)
```

size 的有效输入大体可分为 3 种：

- *n*：读取前 *n* 个整数，并写入一个列向量中；
- *inf*：读至文件末尾；
- [*m*,*n*]：读取数据到 *m*×*n* 的矩阵中，按列排序，*n* 可以是 *inf*，*m* 不可以。

例如，对 t_b.m 文件进行操作，示例代码设置如下：

```
>> fid=fopen('t_b.m','r');
>> data=fread(fid,4)
```

```
data =
     97
     61
     91
     49
>> data1=fread(fid,[3,2])
data1 =
     53     48
     44     44
     50     50
```

其中，后一个 fread 命令读取的数据是紧接第一个 fread 的。

fread 命令还有第 3 个输入变量，用来控制二进制数据转换成 MATLAB 矩阵时所用的精度，命令格式为：

```
a=fread(fid,size,precision)
```

precision 包括两部分：一是数据类型定义，比如 int、float 等；二是一次读取的位数。默认情况下是 uchar（8 位字符型），常用的精度在表 9-1 中有简单的介绍，并且与 C 和 Fortran 中相当的形式做了对比。

表 9-1　　精度类型

MATLAB	C 或 Fortran	描述
'uchar'	'unsigned char'	无符号字符型
'schar'	'signed char'	带符号字符型（8 位）
'int8'	'integer*1'	整型（8 位）
'int16'	'integer*2'	整型（16 位）
'int32'	'integer*4'	整型（32 位）
'int64'	'integer*8'	整型（64 位）
'uint8'	'integer*1'	无符号整型（8 位）
'uint16'	'integer*2'	无符号整型（16 位）
'uint32'	'integer*4'	无符号整型（32 位）
'uint64'	'integer*8'	无符号整型（64 位）
'single'	'real*4'	浮点数（32 位）
'float32'	'real*4'	浮点数（32 位）
'double'	'real*8'	浮点数（64 位）
'float64'	'real*8	浮点数（64 位）

还有一些类型是与平台有关的，平台不同可能位数不同，如表 9-2 所示。

表 9-2　　与平台有关的精度类型

MATLAB	C 或 Fortran	描述
'char'	'char*1'	字符型(8 位，有符号或无符号)
'short'	'short'	整型（16 位）
'int'	'int'	整型（32 位）
'long'	'long'	整型（32 位或 64 位）
'ushort'	'unsigned short'	无符号整型（16 位）
'uint'	'unsigned int'	无符号整型（32 位）
'ulong'	'unsigned long'	无符号整型（32 位或 64 位）
'float'	'float'	浮点数（32 位）

如果 fid 表示的是已打开的浮点类型的数据文件，则得到下面的代码：

```
>> a=fread(fid,10,'float')
```

将读入前 10 个浮点型数据到列向量 *a* 中。

> 注意：fread 还有两个输入参数，即 *skip* 和 *machineformat*，均不常用，其用法可参考 MATLAB 中 fread 的 help 文件。

9.3　写入二进制文件

函数 fwrite 的作用是将一个矩阵的元素按所定的二进制格式写入某个打开的文件，并返回成功写入的数据个数。格式为：

```
count=fwrite(fid,a,precision)
```

其中，*fid* 是从 fopen 得到的文件标识，*a* 是待写入的矩阵，*precision* 设定了结果的精度，可用的精度类型见 fread 中的叙述。示例代码设置如下：

```
>> fid=fopen('tob.bin','w');
>> count=fwrite(fid,magic(5),'int32');
>> status=fclose(fid)
```

上面的程序段的执行结果是生成一个文件名为 tob.bin 的二进制文件，长度为 100 字节，包含 5×5 个数据，即 5 阶方阵的数据，每个数据占用 4 个字节的存储单位，数据类型为整型，输出变量 *count* 值为 25。由于是二进制文件，所以无法用 type 命令来显示文件内容，如果要查看，可用以下命令：

```
>> fid=fopen('tob.bin','r')
fid =
     4
>> data=(fread(fid,25,'int32'))'
data =
     17     23      4     10     11     24      5      6     12     18      1      7     13     19
25      8     14     20     21      2     15     16     22      3      9
```

9.4　读取文本文件

如果需要将文本文件中的某一行读出，并将该行的内容以字符串形式返回，可采用以下两个命令：

```
tline=fgetl(fid)
tline=fgets(fid)
```

两个函数的功能很相似，均可从文件中读取一行数据，区别在于 fgetl 会舍弃换行符，而 fgets 则保留换行符。示例代码设置如下：

```
fid=fopen('fgetl.m');
```

```
        while 1
            tline = fgetl(fid);
            if ~ischar(tline)
                break;
            end
            disp(tline)
        end
        fclose(fid);
```

命令执行结果为：

```
function tline = fgetl(fid)
%FGETL Read line from file, discard newline character.
%    TLINE = FGETL(FID) returns the next line of a file associated with file
%    identifier FID as a MATLAB string. The line terminator is NOT
%    included. Use FGETS to get the next line with the line terminator
%    INCLUDED. If just an end-of-file is encountered then -1 is returned.
%    FGETL is intended for use with text files only.    Given a binary file
%    with no newline characters, FGETL may take a long time to execute.
%    Example
%        fid=fopen('fgetl.m');
%        while 1
%            tline = fgetl(fid);
%            if ~ischar(tline), break, end
%            disp(tline)
%        end
%        fclose(fid);%
%    See also FGETS, FOPEN.
%    Copyright 1984-2003 The MathWorks, Inc.
%    $Revision: 5.15.4.2 $    $Date: 2004/04/10 23:29:22 $
try
    [tline,lt] = fgets(fid);
    tline = tline(1:end-length(lt));
    if isempty(tline)
        tline = '';
    end
catch
    if nargin ~= 1
        error (nargchk(1,1,nargin,'struct'))
    end
    if isempty(fopen(fid))
        error ('MATLAB:fgetl:InvalidFID','Invalid file identifier.')
```

```
        end
        rethrow(lasterror)
    end
```

若已知 ASCII 文件的格式，要进行更精确的读取，可用 fscanf 函数从文件中读取格式化的数据，其使用语法如下：

```
    [a,count]=fscanf(fid,format,size)
```

此命令从文件标识为 *fid* 的文件中读取数据，并转换成指定的 *format* 格式字符串，返回到矩阵 *a* 中。*count* 是可选输出项，表示成功读取的数据个数。*size* 是可选输入项，它对可以从文件读取的数据数目做了限制，如果没有指定，则默认为整个文件，否则可以指定为 3 种类型：*n*，*inf*，[*m*,*n*]。各项的意义同 9.2 节中的描述。

format 用于指定读入数据的类型，常用的格式有：

- %s：按字符串进行输入转换；
- %d：按十进制数据进行转换；
- %f：按浮点数进行转换。

另外还有其他的格式，它们与 C 语言中的 fprintf 中参数的用法是相同的，可以参阅 C 手册。

在格式说明中，除了单个的空格字符可以匹配任意个数的空格字符外，通常的字符在输入转换时将与输入的字符一一匹配，函数 fscanf 将输入的文件看作是一个输入流，MATLAB 根据格式来匹配输入流，并将在流中匹配的数据读入到 MATLAB 系统中。下面是一个文件输入的例子：

```
    >> fid=fopen('table.txt','r');
    >> title=fscanf(fid,'%s');
    >> status=fclose(fid);
```

读取的数据解释为字符串，结果为：

```
    title =
    1491215262034234676
```

用 type 命令读取 table.txt 文件内容为：

```
    >> type table.txt
    1 4 9 12 15 26
    20 34 23 46 76
```

如果以双精度格式来读取的话，则程序和结果为：

```
    >> fid=fopen('table.txt','r')
    >> data=fscanf(fid,'%f')
    data =
         1
         4
         9
        12
        15
        26
```

```
        20
        34
        23
        46
        76
```

上例显示了 MATLAB 的 fscanf 函数和 C 中 fscanf 函数的不同之处。MATLAB 的 fscanf 函数是向量化的，只要读入数据的类型正确，MATLAB 的 fscanf 函数会一再执行，并把所得结果以向量形式返回。

如果要对返回向量的形式和大小进行限制，可以加入 *size* 输入变量。示例代码设置如下：

```
>> fid=fopen('table.txt','r',[3,4]);
>>data=fscanf(fid,'%f',[3,4]);
>> status=fclose(fid);
data =
     1    12    20    46
     4    15    34    76
     9    26    23     0
```

> 注意：（1）fscanf 在读取文件时，是逐行进行的，在返回矩阵时，是将数据逐列写入的。（2）本节所介绍的命令不能对二进制文件进行操作。（3）sscanf 函数和 fscanf 函数的功能类似，不同的是 sscanf 是从字符串中读取数据，而不是对文件的操作。

9.5 写入文本文件

MATLAB 的函数 fprintf 的作用是将数据转换成指定格式字符串，写入到文本文件中。其语法格式为：

```
count=fprintf(fid,format,y)
```

其中 *fid* 是要写入的文件标识，由 fopen 产生，*format* 是格式指定字符串，用以指定数据写至文件的格式，*y* 是 MATLAB 的数据变量。*count* 是返回的成功写入的字节数。

fid 值也可以是代表标准输出的 1 和代表标准出错的 2，如果 *fid* 字段省略，则默认值为 1，输出到屏幕上。常用的格式类型说明符如下：

- %e：科学计数形式，即数值表示成 $a \times 10^b$ 形式；
- %f：固定小数点位置的数据形式；
- %g：在上述两种格式中自动选取较短的格式。

可以用一些特殊格式比如\n、\r、\t、\b、\f 等来产生换行、回车、tab、退格、走纸等字符，用\\来产生反斜线符号\，用%%来产生百分号。此外还可以包括数据占用的最小宽度和数据精度的说明。举例说明此命令的用法：

```
%将一个平方根表写入 s_table.dat
a=1:10;
b=[a;sqrt(a)];
```

```
fid=fopen('s_table.dat','w');
fprintf(fid,'table of square root:\n');
fprintf(fid,'%2.0f    %6.4f\n',b);
fclose(fid);
```

运行后将 s_table.dat 打印出来，可得如下代码：

```
>> type s_table.dat
table of square root:
 1    1.0000
 2    1.4142
 3    1.7321
 4    2.0000
 5    2.2361
 6    2.4495
 7    2.6458
 8    2.8284
 9    3.0000
10    3.1623
```

在本例中，第一条 fprintf 语句输出一行标题，随后换行，第二条 fprintf 语句输出函数值表，每组自变量和函数占一行，都是固定小数点位置的形式。自变量值占 2 个字符位，不带小数；函数值占 6 个字符位，小数点后的精度是 4 个字符位，自变量和函数值之间空两格。

矩阵 *y* 的元素按列的顺序转换成格式化的输出，函数反复使用格式说明，直至将矩阵的数据转换完毕。

> 注意：sprintf 函数与 fprintf 函数功能类似，但是 sprintf 将数据以字符串形式返回，而不是直接写入文件。

9.6　文件内的位置控制

根据操作系统的规定，在读写数据时，缺省的方式总是从磁盘文件的开始顺序向后的在磁盘空间上读写数据。操作系统通过一个文件指针，来指示当前的文件位置。C 或 Fortran 语言都有专门的函数来控制和移动文件指针，达到随机访问磁盘文件的目的。MATLAB 中也有类似的函数，如表 9-3 所示。

表 9-3　　控制文件内位置指针的函数

函数	功能
feof	测试指针是否在文件结束位置
fseek	设定文件指针位置
ftell	获取文件指针位置
frewind	重设指针至文件起始位置

- feof

feof 用于测试指针是否在文件结束位置，其语法格式为：

```
feof(fid)
```

如果文件标识为 *fid* 的文件的末尾指示值被置位，则此命令返回 1，说明指针在文件末尾，否则返回 0。

- fseek

fseek 用于设定指针位置，其语法格式为：

```
status=fseek(fid,offset,origin)
```

其中 *fid* 是文件标识；*offset* 是偏移量，以字节为单位，可以是整数（表示要往文件末尾方向移动指针）、0（不移动指针位置）或负数（表示往文件起始方向移动指针）；*origin* 是基准点，可以是'bof'（文件的起始位置）、'cof'（指针的目前位置）、'eof'（文件的末尾），也可以用-1、0 或 1 来表示。

如果返回值 *status* 为 0 表示操作成功，返回-1 表示操作失败，如果要了解更多信息可以调用 ferror 函数。

- ftell

ftell 用于返回现在的位置指针，其语法格式为：

```
position=ftell(fid)
```

返回值 *position* 是距离文件起始位置的字节数，如果返回-1 则说明操作失败。

- frewind

frewind 用于将指针返回到文件开始，语法格式为：

```
frewind(fid)
```

下面通过一个例子来介绍这几个命令的使用方法，示例代码如下：

```
a=[1:6];
fid=fopen('six.bin','w');
fwrite(fid,a,'short');
status=fclose(fid);
fid=fopen('six.bin','r');
six=fread(fid,'short');
eof=feof(fid);
frewind(fid);
status=fseek(fid,2,0);
position=ftell(fid);
```

运行后，检查 *six*、*eof*、*status* 和 *position* 的值：

```
>> six'
ans =
     1     2     3     4     5     6
>> eof
eof =
     1
>> status
```

```
status =
        0
>> position
position =
        2
```

第 10 章　信号处理工具箱

MATLAB 的推出得到了各个领域专家学者的关注，其强大的扩展功能为各个领域的应用提供了基础，相继推出了各种 MATLAB 工具箱，为各个层次的研究人员提供了有力的工具，节省了时间。本章将介绍信号处理（signal processing）工具箱的使用。

10.1　数字信号处理基本理论

在计算机中，所有的信号都是离散信号，因此在使用 MATLAB 进行信号处理之前，首先要了解离散时间信号处理的相关理论。

10.1.1　离散信号与系统

1．离散信号

信号是信息的表现形式，是通信传输的客观对象，其特性可以从两个方面来描述，即时间特性和频率特性。信号之所以不同是因为各自有不同的时间特性和频率特性，并且两者之间有一定的对应关系。若 t 是定义在时间轴上的离散点，则 $x(t)$为离散时间信号，表示为 $x(nT)$，T 表示相邻两个点之间的时间间隔，也称为抽样周期，n 取整数并代表时间的离散时刻。一般可以把 T 归一化为 1，这样 $x(nT)$可简记为 $x(n)$，又称为离散时间序列。

在 MATLAB 中，可用一个向量来表示一个有限长度的序列（由于内存的限制，不可能表示一个任意无限序列），示例代码设置如下：

```
x(n)=[4,5,2,3,9,8,0,7,-1]
```

MATLAB 对下标的约定从 1 开始递增，如果要包含采样时刻的信息，则需要两个向量来表示，对于上面的序列应有：

```
n=[-2,-1,0,1,2,3,4,5,6]                    x=[4,5,2,3,9,8,0,7,-1]
```

为了分析的方便，数字信号处理理论中定义了一些基本的典型的序列，其定义和 MATLAB 表达式如下：

- 单位取样序列

$$d(n)=\begin{cases}1 & n=0\\ 0 & n\neq 0\end{cases}$$

该表达式可利用 zeros 函数或者逻辑关系来实现，代码设置：

```
x=zeros(1,N);
x(1)=1;
```

或者可以写成：

```
n=1:N
x=[n==1];
```

- 单位阶跃序列

$$u(n)=\begin{cases}1 & n\geq 0\\ 0 & n<0\end{cases}$$

该表达式可利用 ones 函数来实现，代码设置如下：

```
x=ones(1,N)
```

- 实指数序列

$$x(n)=a^{n}\quad \forall n;\ a\in \mathrm{R}$$

该表达式通过 MATLAB 实现，代码设置如下：

```
n=0:N-1;
x=a.^n;
```

- 复指数序列

$$x(n)=e^{(s+jw_0)n}\quad \forall n$$

该表达式通过 MATLAB 实现，代码设置如下：

```
n=0:N-1;
x=exp(a+j*w0)*n);
```

- 正（余）弦序列

$$x(n)=\cos(w_0 n+q)\quad \forall n$$

该表达式通过 MATLAB 实现，代码设置如下：

```
n=0:N-1;
x=cos(w0*n+q);
```

- 随机序列

在 MATLAB 中提供了两类（伪）随机信号：

rand(1,N)产生[0,1]上均匀分布的随机矢量；randn(1,N)产生均值为 0，方差为 1 的高斯随机序列。其他分布的随机数可通过上述随机数的变换而产生。

- 周期序列

x(n)=x(n+N)　　$\forall n$

2．离散系统

系统是由若干相互作用和相互依赖的事物组合而成的并具有特定功能的整体。系统分析的着眼点是分析系统的输入和输出的关系，而不涉及系统的内部情况。

一个离散系统可以抽象为一种变换或者一种映射，即 y(n)=*T*[x(n)]。其中 *T* 代表变换。在数字信号处理中，所研究的系统基本上都是线性时不变系统（LSI 系统），其输入/输出关系可以通过冲激响应 *h(n)*表示：

$$y(n)=x(n)\otimes h(n)=\sum_{k=-\infty}^{\infty}x(k)h(n-k)$$

其中 *x(n)*和 *h(n)*作卷积运算，MATLAB 提供了求卷积函数 conv。有关线性时不变系统中涉及的重要概念，比如稳定性、因果性和移不变性等可参看信号与系统的经典教材。

10.1.2 Z 变换

Z 变换是离散系统和离散信号分析、综合的重要工具，其地位和作用如同拉普拉斯变换之于连续信号和系统。

1. Z 变换的定义

给定离散信号 *x(n)*，其 Z 变换的定义为：

$$X(z)=\sum_{n}x(n)z^{-n}$$

其中 z 为复变量。如果 n 的取值范围为 $-\infty\sim+\infty$，则上式定义的 Z 变换称为双边 Z 变换，如果 n 的取值范围为 $0\sim+\infty$，则上式称为单边 Z 变换。实际的物理系统的抽样响应 $h(n)$ 在 $n<0$ 时恒为 0，因此对应的都是单边 Z 变换。

$X(z)$存在的 z 的集合称为收敛域(ROC)，即满足下式：

$$X(z)=\sum_{n}x(n)z^{-n}<\infty$$

ROC 由$|z|$决定，因此一般为环形。序列在单位圆上的 Z 变换就是序列的傅立叶变换，若 $x(n)=h(n)$，则该变换就是系统的频率响应。

2. Z 变换的性质

- 线性

若 $Z[x_1(n)]=X_1(z)$，$ROC:R_1$；$Z[x_2(n)]=X_2(z)$，$ROC:R_2$。则有：

$Z[ax_1(n)+bx_2(n)]=aX_1(z)+bX_2(z)$，ROC 为 R_1 和 R_2 的公共部分。

- 时移性质

$Z[x(n-k)]=z^{-k}X(z)$，ROC 不变。

- 频移性质

$Z[a^n x(n)]=X(\frac{z}{a})$，$ROC:ROC_x/|a|$

- 折叠

$Z[x(-n)]=X(\frac{1}{z})$，$ROC:$与ROC_x相反，即ROC_x的逆。

- 复共轭

$Z[x^*(n)] = X^*(z^*)$， $ROC: ROC_x$

- Z 域微分

$Z[nx(n)] = -z\dfrac{dX(z)}{dz}$， $ROC: ROC_x$

- 序列相乘

$Z[x_1(n)x_2(n)] = \dfrac{1}{2pj}\oint_c X_1(v)X_2(\dfrac{z}{v})v^{-1}dv$，$ROC: ROC_{x_1} \cap ROC_{x_2}$的逆，其中 C 为公共 ROC 中包围原点的闭合围线。

- 序列卷积

$Z[x_1(n) \otimes x_2(n)] = X_1(z)X_2(z)$， $ROC: ROC_{x_1} \cap ROC_{x_{21}}$

10.1.3　离散傅立叶变换

Z 变换提供了任意序列在频域的表示方法，但它是连续变量 *z* 的函数，因此无法直接利用计算机进行数值计算。为了使用 MATLAB，必须截断序列，得到有限个点的表达式。这就产生了离散傅立叶级数（DFS）、离散傅立叶变换（DFT）和计算量小的快速傅立叶变换（FFT）。

如果信号在频域上是离散的，则该信号在时域上就是周期性的函数。反之在时域上离散的信号在频域上必然表现为周期性的频率函数。可以得出一个一般规律：一个域的离散必然造成另一个域的周期延拓。这种离散变换，本质上都是周期的。因此，首先从周期序列及其傅立叶级数开始讨论，然后再讨论可作为周期序列一个周期的、有限长序列的离散傅立叶变换（DFT）。

1．**离散傅立叶级数**（DFS）

对于周期为 *N* 的离散时间信号序列 $\tilde{x}(n) = \tilde{x}(n+kN)$，其中 *k* 为任意整数，由于在 *Z* 平面上没有任何收敛区域，所以不能进行 *Z* 变换，但是可以用傅立叶级数来表达，其基波频率为 $2p/N$，用复指数表示为 $e_1(n) = e^{j\frac{2p}{N}n}$，第 *k* 次谐波为 $e_k(n) = e^{j\frac{2p}{N}kn}$，所以有 $e_{k+N}(n) = e_k(n)$，可得离散傅立叶级数公式如下：

$$\tilde{x}(n) = \frac{1}{N}\sum_{k=0}^{N-1}\tilde{X}(k)e^{j\frac{2p}{N}kn}$$

在上式中求和号前所乘的系数 1/*N* 是习惯上采用的常数，$\tilde{X}(k)$ 是 *k* 次谐波的系数：

$$\tilde{X}(k)=\sum_{n=0}^{N-1}\tilde{x}(n)e^{-j\frac{2p}{N}kn}$$

习惯上也常采用符号$W_N=e^{-j(2p/N)}$，这样离散傅立叶级数对可以表示为：

$$\begin{cases}\tilde{x}(n)=\dfrac{1}{N}\sum\limits_{k=0}^{N-1}\tilde{X}(k)W_N{}^{-kn}\\ \tilde{X}(k)=\sum\limits_{n=0}^{N-1}\tilde{x}(n)W_N{}^{kn}\end{cases}$$

2．离散傅立叶变换（DFT）

周期序列实际上只有有限个序列值有意义，因此它的许多特性可以沿用到有限长序列上，对于一个长度为 N 的有限长序列 $x(n)$，以 $x(n)$为主值序列并以 N 为周期进行延拓得到周期序列$\tilde{x}(n)$，即$\tilde{x}(n)=\sum\limits_{r=-\infty}^{\infty}x(n+rN)$，当 0≤$n$≤$N$-1 时，$x(n)=\tilde{x}(n)$，$n$ 为其余值时 $x(n)$为 0。

由离散傅立叶级数公式，可以得到有限长序列 $x(n)$的离散傅立叶变换公式：

$$x(n)=\frac{1}{N}\sum_{k=0}^{N-1}X(k)W_N{}^{-kn}\qquad 0\leqslant n\leqslant N-1$$

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N{}^{kn}\qquad 0\leqslant k\leqslant N-1$$

离散傅立叶级数和离散傅立叶变换与 Z 变换有类似的性质，详细的介绍可以参照信号与系统教材，本节不再赘述。

3．快速傅立叶变换（FFT）

快速傅立叶变换是离散傅立叶变换的快速算法，其应用极大地推动了 DSP 理论和技术的发展。下面介绍 FFT 的基本思想和实现。

- FFT 的基本思想

N 点序列的 DFT 为

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N{}^{kn}\qquad 0\leqslant k\leqslant N-1$$

由于系数$W_N{}^{kn}$是一个周期函数：$W_N{}^{n(N-k)}=W_N{}^{k(N-n)}=W_N{}^{-nk}$，且是对称的：$W_N{}^{nk+N/2}=-W_N{}^{nk}$。

FFT 正是基于这样的基本思想而发展起来的。

- FFT 的实现方法

FFT 的算法形式有很多种，但基本上可以分为两大类：时间抽取法（DIT-FFT）和频率抽取法（DIF-FFT）。这两种算法在思想上基本一致，只是划分方式略有差异，这里仅以 DIT-FFT 为例进行说明。

当 N 是 2 的整数次方时，称为基 2 的 FFT 算法。

首先将序列 $x(n)$分解为两组，偶数项为一组，奇数项为一组：

$$\begin{cases} x(2r) = x_1(r) \\ x(2r+1) = x_2(r) \end{cases} \qquad r=0,1,\cdots,N/2-1$$

将两组分别进行 $N/2$ 点的 DFT 可得：

$$\begin{cases} X(k) = X_1(k) + W_N{}^k X_2(k) \\ X(N/2+k) = X_1(k) - W_N{}^k X_2(k) \end{cases} \qquad k=0,1,\cdots,N/2-1$$

重复这一过程，可得到 $x(n)$的 FFT，整个运算需要 $\frac{N}{2}\log_2 N$ 次复数乘法和 $N\log_2 N$ 次复数加法。

在 MATLAB 中，可直接利用内部函数 fft 进行计算，速度比较快，有关其具体用法在本章的第 3 节会给出例子。

10.1.4　数字滤波器结构

数字信号处理的目的之一是设计某种设备或建立某种算法用以处理序列，使序列具有某种确定的性质，这种设备或算法结构就称为数字滤波器。数字滤波器可以分为有限冲激响应（FIR）和无限冲激响应（IIR）两种。滤波器的设计结果受滤波器的类型和结构的影响。本节分别介绍这两种滤波器的结构。

1．IIR 滤波器

IIR 滤波器的系统函数为：

$$H(z) = \frac{\sum_{r=0}^{M} b_r z^{-r}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

其差分方程可表示为：

$$y(n) = \sum_{r=0}^{M} b_r x(n-r) - \sum_{K=1}^{N} a_k y(n-k)$$

由其系统函数或差分方程可知，实现 IIR 滤波器有直接型、级联型和并联型三种结构。

- 直接型

直接型分为直接Ⅰ型和直接Ⅱ型两种，直接Ⅰ型可由差分方程得到，第一部分 $\sum_{r=0}^{M} b_r x(n-r)$ 表示将输入信号进行延时，组成 M 节的延时网络，把每节延时抽头与常系数相乘，再把结果相加，这是一个横向结构网络，即实现零点的网络，第二部分表示将信号输出延时，组成 N 节的延时网络，把每节延时抽头后与常系数相乘，然后再把结果相加，由于这部分是对输出的延时，故为反馈网络，这部分网络实现极点。$y(n)$的信号流图如图 10-1 所示。

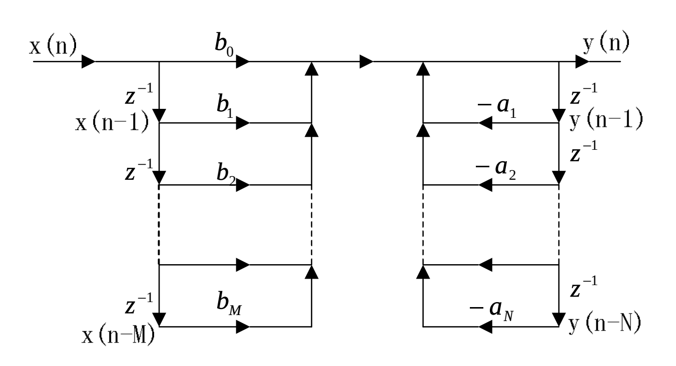


图 10-1　直接Ⅰ型

直接Ⅱ型可以由系统函数得到，将分子和分母的倒数分别看作一个网络，则整个系统可看作是这两个网络的级联，如图 10-2 所示。

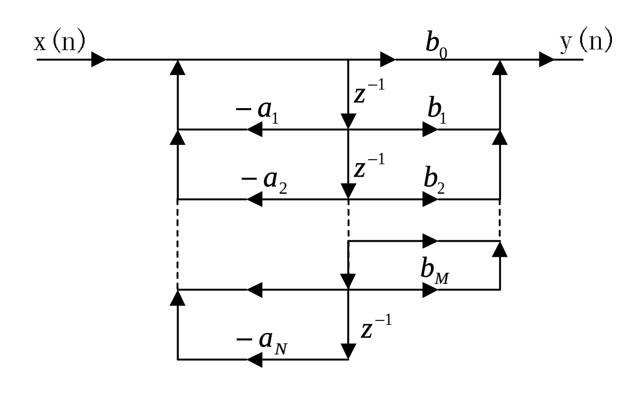


图 10-2　直接Ⅱ型

直接Ⅱ型结构比直接Ⅰ型速度快且经济（延时单元少一半），但它们的系数都不能直接控制滤波器特性，对字长效应敏感。

在 MATLAB 中，可以 DSP 工具函数 filter 实现 IIR 的直接形式。

- 级联型

对 IIR 滤波器系统函数进行因式分解，若 a_k, b_r 均为实数，则零点和极点只有实数和共轭复数，合并共轭因子有如下的表达式：

$$H(z)=\frac{A\prod_{i=1}^{M_1}(1-g_i z^{-1})\prod_{i=1}^{M_2}(1+b_{1i}z^{-1}+b_{2i}z^{-2})}{\prod_{i=1}^{N_1}(1-p_i z^{-1})\prod_{i=1}^{N_2}(1+a_{1i}z^{-1}+a_{2i}z^{-2})}$$

将单实根视为二阶因子的特例，设 $M\leqslant N$，则：

$$H(z)=A\prod_{i=1}^{L}\frac{(1+b_{1i}z^{-1}+b_{2i}z^{-2})}{(1+a_{1i}z^{-1}+a_{2i}z^{-2})}=A\prod_{i=1}^{L}H_i(z)$$

其中，L 表示 $(N+1)/2$ 中的最大整数。$H_i(z)$ 称为二阶基本节，可以采用直接 II 型实现，

$H(z)$ 则由 $H_i(z)$ 级联而成，级联型的具体体现形式如图 10-3 所示。

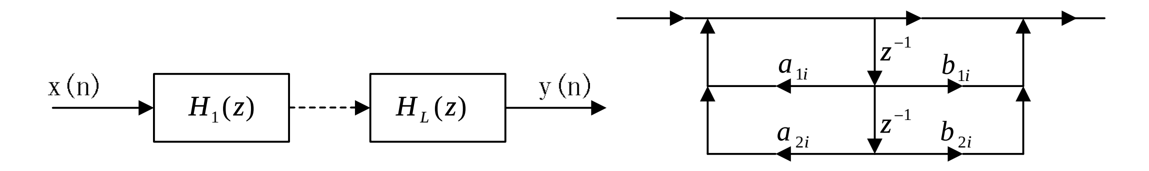


图 10-3　级联型

在 MATLAB 中，给定直接形式滤波器的系数，可计算出级联形式的系数，这可由扩展函数 dir2cas 实现，然后利用扩展函数 casfiltr 实现滤波器的级联形式。

- 并联型

作为系统函数的另一种表示形式，可以将 H(z)表示如下形式的部分分式展开：

$$H(z)=\sum_{k=0}^{M-N}G_k z^{-k}+\sum_{k=1}^{K}\frac{b_{0k}+b_{1k}z^{-1}}{1+a_{1k}z^{-1}+a_{2k}z^{-2}}$$

其中 $K=N/2$，每一部分均为二阶，它们之间是并联关系，$M=N=4$ 为例，画出其结构图如图 10-4 所示。

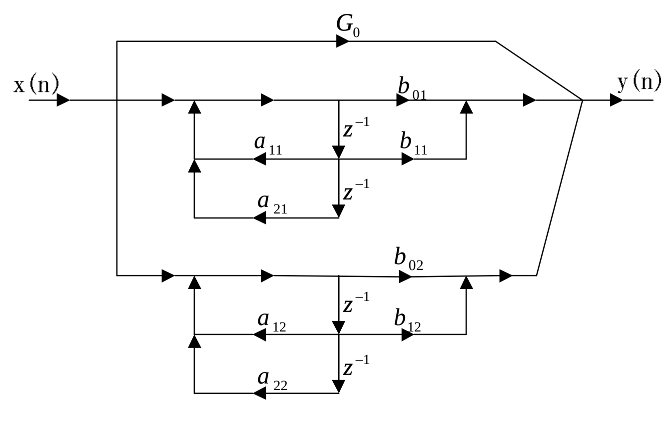


图 10-4　并联型

在 MATLAB 下，已经设计了将滤波器直接形式转化为并联形式的扩展函数 dir2par，其中用到了复共轭对比较函数 cplxcomp，这样就可利用 parfiltr 函数实现滤波器的并联形式。

2．FIR 滤波器

有限冲激响应（FIR）滤波器的系统函数为：

$$H(z)=b_0+b_1z^{-1}+\ldots+b_{M-1}z^{-(M-1)}$$

差分方程表示为：

$$y(n)=b_0x(n)+b_1x(n-1)+\ldots+b_{M-1}x(n-M+1)$$

与 IIR 滤波器相比，FIR 滤波器结构相对简单，而且可以设计成线性相位。实现 FIR 滤波器可采用直接形式、级联形式、线性相位形式和频率取样形式四种结构。

- 直接形式

与 IIR 类似，以 M=5 为例，结构如图 10-5 所示，在 MATLAB 中可通过 filter 函数实现。

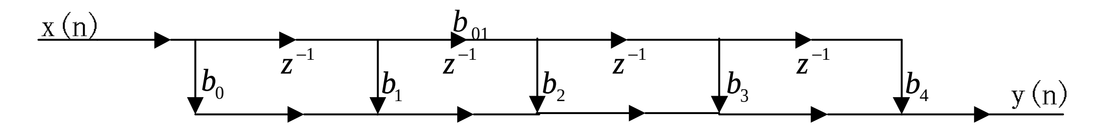


图 10-5　FIR 滤波器的直接形式

- 级联形式

$$H(z)=b_0+b_1z^{-1}+\ldots+b_{M-1}z^{-(M-1)}=b_0\prod_{k=1}^{K}(1+b_{k1}z^{-1}+b_{k2}z^{-2})$$

其中，K=M/2，以 M=7 为例，其结构如图 10-6 所示。

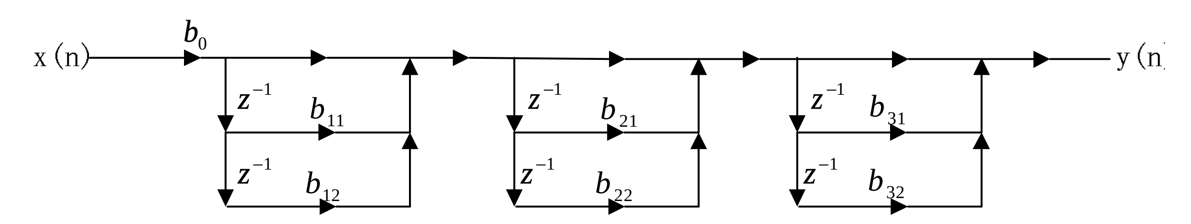


图 10-6　FIR 滤波器级联形式

在 MATLAB 中，可采用 dir2cas 和 casfiltr 来实现。

- 线性相位形式

线性相位条件 $\angle H(e^{jw})=b-aw$，$-p<w\leqslant p$，其中 $b=0$或 $\pm p/2$，a 为常数。

对 $H(z)$则有下列对称性：

$$h(n)=h(M-1-n)\qquad b=0,\quad 0\leqslant n\leqslant M\text{-}1$$　（对称冲激响应）

$$h(n) = -h(M-1-n) \qquad b = \pm\frac{p}{2}，0 \leqslant n \leqslant M\text{-}1 \qquad （反对称冲激响应）$$

以 M 为 7（代表奇数）和 M 为 6（代表偶数）为例，对称型的结构如图 10-7 所示。

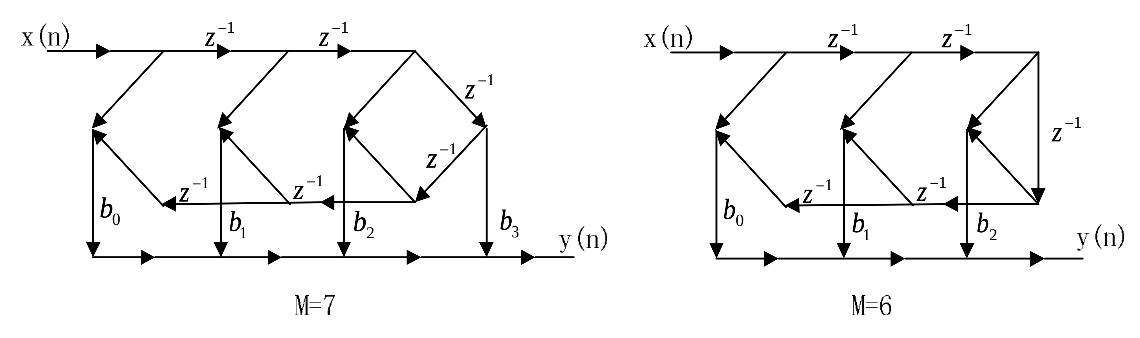


图 10-7　FIR 滤波器线性相位形式

在 MATLAB 中，实现 FIR 线性相位形式的函数等同于直接形式，可用 filter 函数。

- 频率取样形式

冲激响应 h(n)的 M 点 DFT 为 H(k)（$0 \leqslant k \leqslant M$-1），则有 H(z)=Z[h(n)]=Z[IDFT(H(k))]
利用内插公式可得：

$$H(z) = \left(\frac{1-z^{-M}}{M}\right)\sum_{k=0}^{M-1}\frac{H(k)}{1-W_M{}^{-k}z^{-1}}$$

这表明在这种结构中采用了离散傅立叶变换 $H(k)$，而不是冲激响应 $h(n)$。这种结构分成两部分，以 M 取 4 为例，频率取样形式如图 10-8 所示。

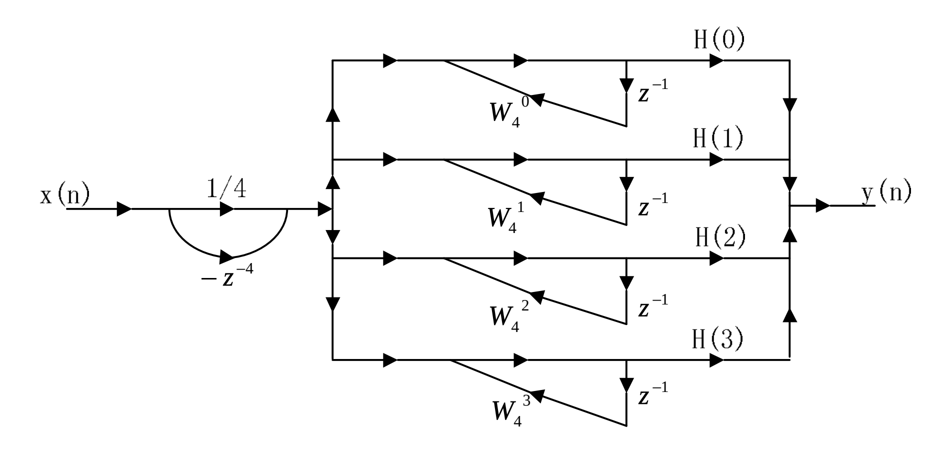


图 10-8　FIR 滤波器频率取样形式

在 MATLAB 中，给定 $h(n)$或 $H(k)$，则可借助于扩展函数 dir2fs 得到频率取样形式，它将 $h(n)$值转换为频率取样形式。

此外还有一种广泛应用的格形滤波器结构，可参阅有关文献。在第 3 节中将具体讨论这些滤波器基于 MATLAB 的实现方法。

10.2 MATLAB 7.0 的信号处理工具箱函数

MATLAB 包含了进行信号处理的许多工具箱函数，有关这些工具箱函数的使用可通过 Help 命令得到。为使用方便，本节将按分组给出这些函数的简单说明，在后面的章节会给出其中一部分函数的使用举例。详细的文档可以在 Help 目录下找到。

10.2.1 波形产生（Waveform Generation）

在该工具箱里包含以下几个函数：

- chirp: 产生扫频余弦函数;
- diric: 产生 Dirichlet 或周期 sinc 函数;
- gauspuls: 产生高斯调制的正弦曲线脉冲;
- gmonopuls: 产生高斯单脉冲;
- pulstran: 产生一个脉冲序列;
- rectpuls: 产生一个非周期的抽样方波;
- sawtooth: 产生锯齿波或三角波;
- sinc: 产生 sinc 函数，即 $\dfrac{\sin(pt)}{pt}$;
- square: 产生方波;
- tripuls: 产生一个非周期的采样三角波;
- vco: 压控振荡器。

10.2.2 滤波器分析（Filter Analysis）

在该工具箱里包含以下几个函数：

- abs: 求绝对值（幅值，这是一个 MATLAB 函数）;
- angle: 求相角（这是一个 MATLAB 函数）;
- freqs: 模拟滤波器的频率响应;
- freqspace: 频率响应中的频率间隔（这是一个 MATLAB 函数）;
- freqz: 计算数字滤波器的频率响应;
- fvtool: 打开滤波器可视化工具;
- grpdelay: 计算平均滤波器延迟（群延迟）;
- impz: 计算数字滤波器的冲激响应;
- phasedelay: 计算数字滤波器的相位延迟响应;
- phasez: 计算数字滤波器的相位响应;
- stepz: 计算数字滤波器的阶跃响应;
- unwrap: 展开相角（这是一个 MATLAB 函数）;
- zerophase: 计算数字滤波器的零相位响应;
- zplane: 离散系统零极点图。

10.2.3　滤波器实现（Filter Implementation）

在该工具箱里包含以下几个函数：

- conv：求卷积和多项式乘法（这是一个 MATLAB 函数）;
- conv2：二维卷积（这是一个 MATLAB 函数）;
- convmtx：卷积矩阵;
- deconv：反卷积和多项式除法（这是一个 MATLAB 函数）;
- fftfilt：采用重叠相加法基于 FFT 的 FIR 滤波器实现;
- filter：直接滤波器实现（这是一个 MATLAB 函数）;
- filter2：二维数字滤波（这是一个 MATLAB 函数）;
- filtfilt：零相位数字滤波;
- filtic：直接 II 型滤波器的初始条件选择;
- latcfilt：格型和格－梯型滤波器实现;
- medfilt1：一维中值滤波;
- sgolayfilt：Savitzky-Golay 滤波;
- sosfilt：二阶（四次）IIR 数字滤波;
- upfirdn：过采样，FIR 滤波和抽样。

10.2.4　线性系统变换（Linear System Transformations）

在该工具箱里包含以下几个函数：

- latc2tf：将格形滤波器参数转换维传输函数格式;
- polystab：稳定多项式;
- polyscale：多项式的根的数值范围;
- residuez：Z 变换部分分式展开或留数计算;
- sos2ss：变系统二阶分割形式为状态空间形式;
- sos2tf：变系统二阶分割形式为传递函数形式;
- sos2zp：变系统二阶分割形式为零极点增益形式;
- ss2sos：变系统状态空间形式为二阶分割形式;
- ss2tf：变系统状态空间形式为传递函数形式;
- ss2zp：变系统状态空间形式为零极点增益形式;
- tf2latc：变传递参数形式为格形滤波器形式;
- tf2sos：变传递函数形式为系统二阶分割形式;
- tf2ss：变传递函数形式为系统状态空间形式;
- tf2zp：变连续时间传递函数为零极点增益形式;
- tf2zpk：变离散时间传递函数为零极点增益形式;
- zp2sos：变零极点增益形式为二阶分割形式;
- zp2ss：变零极点增益形式为状态空间形式;
- zp2tf：变零极点增益形式为传递函数形式。

10.2.5 FIR 滤波器设计（FIR Digital Filter Design）

在该工具箱里包含以下几个函数：

- cfirpm：复杂非线性相位等纹波滤波器设计；
- dfilt：用面向对象的方式产生滤波器；
- fir1：基于窗函数的 FIR 滤波器设计；
- fir2：基于频率取样的 FIR 滤波器设计；
- fircls：多波段有限最小二乘 FIR 滤波器设计；
- fircls1：低通和高通线性相位 FIR 滤波器的有限最小二乘设计；
- firgauss：高斯 FIR 滤波器设计；
- firls：最小二乘线性相位 FIR 滤波器设计；
- firpm：Parks-McClellan 最优化 FIR 滤波器设计；
- firpmord：Parks-McClellan 最优化 FIR 滤波器阶估计；
- firrcos：升余弦 FIR 滤波器设计；
- intfilt：内插 FIR 滤波器设计；
- kaiserord：用 Kaiser 窗进行设计的 FIR 滤波器的参数估计；
- sgolay：Savitzky-Golay 滤波器设计。

10.2.6 IIR 滤波器设计（IIR Digital Filter Design）

在该工具箱里包含以下几个函数：

- butter：Butterworth（巴特沃思）模拟和数字滤波器设计；
- cheby1：Chebyshev（切比雪夫）I 型滤波器设计；
- cheby2：Chebyshev II 型滤波器设计；
- dfilt：用面向对象的方法产生滤波器；
- ellip：椭圆滤波器设计；
- filtstates：包含滤波器状态信息的对象；
- maxflat：归一化数字 Butterworth 滤波器设计；
- yulewalk：递归数字滤波器设计。

10.2.7 IIR 滤波器阶的选择（IIR Filter Order Estimation）

在该工具箱里包含以下几个函数：

- buttord：计算 Butterworth 滤波器的阶和截止频率；
- cheb1ord：计算 Chebyshev I 型滤波器的阶；
- cheb2ord：计算 Chebyshev II 型滤波器的阶；
- ellipord：计算椭圆滤波器的最小阶。

10.2.8　变换（Transforms）

在该工具箱里包含以下几个函数：

- bitrevorder：将输入序列按比特反向变换；
- czt：线性调频 Z 变换；
- dct：离散余弦变换（DCT）；
- dftmtx：离散傅立叶变换矩阵；
- digitrevorder：将输入序列按数字反向变换；
- fft：一维快速傅立叶变换；
- fft2：二维快速傅立叶变换；
- fftshift：重新编排 FFT 函数的输出；
- goertzel：用二阶 Goertzel 算法计算离散傅立叶变换；
- hilbert：希尔伯特变换；
- idct：逆离散余弦变换；
- ifft：一维逆快速傅立叶变换；
- ifft2：二维逆快速傅立叶变。

10.2.9　统计信号处理和谱分析（Statistical Signal Processing and Spectral Analysis）

在该工具箱里包含以下几个函数：

- corrcoef：计算相关系数矩阵；
- corrmtx：计算自相关矩阵的数据矩阵；
- cov：协方差矩阵；
- cpsd：两个信号的互谱密度估计；
- dspdata：DSP 数据对象的参数信息；
- dspopts：频谱对象的可选参数信息；
- mscohere：两个信号之间的幅度自相关函数估计；
- pburg：基于 Burg 方法的功率谱密度估计；
- pcov：基于协方差方法的功率谱密度估计；
- peig：基于特征向量方法的伪谱；
- periodogram：基于周期图的功率谱密度估计；
- pmcov：基于修正协方差方法的功率谱密度估计；
- pmtm：基于 MTM 方法的功率谱密度估计；
- pmusic：基于 MUSIC 算法的功率谱密度估计；
- pwelch：基于 Welch 方法的功率谱密度估计；
- pyulear：基于 Yule-Walker AR 方法的功率谱密度估计；
- rooteig：基于特征向量方法的频率和功率分析；
- rootmusic：基于 root MUSIC 算法的频率和功率分析；
- spectrum：含有频谱估计方法的参数信息的对象；

- tfestimate: 从输入和输出估计传递函数;
- xcorr: 互相关函数估计;
- xcorr2: 二维互相关函数估计;
- xcov: 互协方差函数估计。

10.2.10 窗函数（Windows）

在该工具箱里包含以下几个函数：

- barthannwin: 修正的 Bartlett-Hann 窗;
- bartlett: Bartlett（巴特利特）窗;
- blackman: Blackman（布莱克曼）窗;
- blackmanharris: 最小化 4 阶 Blackman-Harris 窗;
- bohmanwin: Bohman 窗;
- chebwin: Chebyshev 窗;
- flattopwin: 平坦顶部窗;
- gausswin: Gaussian（高斯）窗;
- hamming: Hamming（汉明）窗;
- hann: Hann（汉宁）窗;
- kaiser: Kaiser（凯泽）窗;
- nuttallwin: Nuttall 定义的最小化 4 阶 Blackman-Harris 窗;
- parzenwin: Parzen 窗;
- rectwin: 矩形窗;
- sigwin: 用面向对象方法生成窗;
- triang: 三角窗;
- tukeywin: Tukey 窗;
- window: 窗函数生成;
- wvtool: 窗可视化工具。

10.2.11 参数化建模（Parametric Modeling）

在该工具箱里包含以下几个函数：

- arburg: 基于 Burg 方法的 AR 模型参数估算;
- arcov: 基于协方差方法的 AR 模型参数估算;
- armcov: 基于修正协方差方法的 AR 模型参数估算;
- aryule: 基于 Yule-Walker 方法的 AR 模型参数估算;
- ident: 查看系统识别工具箱文件;
- invfreqs: 模拟滤波器拟合频率响应;
- invfreqz: 离散滤波器拟合频率响应;
- prony: 利用 Prony 法的离散滤波器拟合时间响应;
- stmcb: 利用 Steiglitz-McBride 迭代方法求线性模型。

10.2.12　特殊操作（Specialized Operations）

在该工具箱里包含以下几个函数：

- buffer：将信号向量缓存在数据帧矩阵中；
- cell2sos：将二阶分区的单元序列转换为二阶分区矩阵；
- cplxpair：将复数归成复共轭对；
- demod：通信仿真中的解调；
- dpss：离散椭球体序列（Slepian 序列）；
- dpssclear：清除数据库中的 Slepian 序列；
- dpssdir：Slepian 序列的数据库目录；
- dpssload：从数据库中加载 Slepian 序列；
- dpsssave：保存 Slepian 序列；
- eqtflength：使传输函数分子和分母等长；
- modulate：通信仿真中的调制；
- seqperiod：计算序列周期；
- sos2cell：将二阶分区矩阵转换为单元序列；
- specgram：频谱分析；
- stem：离散数据序列作图；
- strips：条状图；
- udecode：将 2n 进制整型输入解码为浮点数输出；
- uencode：将浮点数输入编码为整型输出。

10.2.13　模拟低通滤波器原型（Analog Lowpass Filter Prototypes）

在该工具箱里包含以下几个函数：

- besselap：Bessel 模拟低通滤波器原型；
- buttap：Butterworth 模拟低通滤波器原型；
- cheb1ap：Chebyshev I 型模拟低通滤波器原型；
- cheb2ap：Chebyshev II 型模拟低通滤波器原型；
- ellipap：椭圆模拟低通滤波器原型。

10.2.14　模拟滤波器设计（Analog Filter Design）

在该工具箱里包含以下几个函数：

- besself：Bessel 模拟滤波器设计；
- butter：Butterworth 模拟数字滤波器设计；
- cheby1：Chebyshev I 型滤波器设计；
- cheby2：Chebyshev II 型滤波器设计；
- ellip：椭圆滤波器设计。

10.2.15　模拟滤波器转换（Analog Filter Transformation）

在该工具箱里包含以下几个函数：

- lp2bp：将低通模拟滤波器转换为带通滤波器；
- lp2bs：将低通模拟滤波器转换为带阻滤波器；
- lp2hp：将低通模拟滤波器转换为高通滤波器；
- lp2lp：改变模拟低通滤波器的截止频率。

10.2.16　滤波器离散化（Filter Discretization）

在该工具箱里包含以下几个函数：

- bilinear：双线性变换法实现模拟到数字的滤波器变换；
- impinvar：脉冲响应不变法实现模拟到数字的滤波器变换。

10.2.17　对数倒谱分析（Cepstral Analysis）

在该工具箱里包含以下几个函数：

- cceps：倒谱分析；
- icceps：逆倒谱分析；
- rceps：实倒谱和最小相位重构。

10.2.18　线性预测（Linear Prediction）

在该工具箱里包含以下几个函数：

- ac2poly：将自相关序列转换为预测多项式；
- ac2rc：将自相关序列转换为反射系数；
- is2rc：将反正弦参数转换为反射系数；
- lar2rc：将对数域比例参数转换为反射系数；
- levinson：Levinson-Durbin 递归算法；
- lpc：计算线性预测系数；
- lsf2poly：将线性谱频率转换为预测系数；
- poly2ac：将预测多项式转换为自相关序列；
- poly2lsf：将预测系数转换为线性谱频率；
- poly2rc：将预测多项式转换为反射系数；
- rc2ac：将反射系数转换为自相关序列；
- rc2is：将反射系数转换为反正弦参数；
- rc2lar：将反射系数转换为对数域比例参数；
- rc2poly：将反射系数转换为预测多项式；
- rlevinson：逆 Levinson-Durbin 递归；
- schurrc：利用自相关序列计算反射系数。

10.2.19　多速信号处理（Multirate Signal Processing）

在该工具箱里包含以下几个函数：

- decimate：降低序列的采样速率;
- downsample：采样速率整数倍下降;
- interp：提高采样速率;
- interp1：一维数据插值;
- resample：按有理数因数改变采样率;
- spline：三次样条函数内插;
- upfirdn：过采样，FIR 滤波，取样;
- upsample：采样速率整数倍提高。

10.2.20　图形用户接口（Graphical User Interfaces）

在该工具箱里包含以下几个函数：

- fdatool：打开滤波器设计和分析工具;
- fvtool：打开滤波器可视化工具;
- sptool：交互式数字信号处理工具（SP 工具）;
- wintool：打开窗函数设计和分析工具;
- wvtool：打开可视窗工具。

10.3　基于 MATLAB 的信号处理系统分析与设计

本节主要介绍 MATLAB 在信号处理领域一些常用的实例，是对前两节基础知识的综合运用。

10.3.1　离散信号与系统的 MATLAB 实现

【例 10.1】离散信号的运算是实现各种信号和系统的基础，本例给出信号基本运算的 MATLAB 实现，包括位移、相加、相乘以及变换等。

给定离散信号为 $x_1(n)$ 和 $x_2(n)$，信号 $y_1(n)=x_1(n-k)$，$y_2(n)=x_1(n+k)$，$x(n)=x_1(n)+x_2(n)$，$y(n)=x_1(n)x_2(n)$，编写程序实现上述运算，并且计算 $x(n)$ 的功率和能量，对 $x(n)$ 进行信号折叠和奇偶分解。

MATLAB 程序代码如下：

```
function [yshift,n]=sig_shift(x,m,n0)
%信号延迟的程序，m 为输入 x 的下标
```

```
%n0 为延迟的单位长度
n=m+n0;
yshift=x;

function [yam,n]=sig_proc(x1,n1,x2,n2,s)
%信号相加和相乘的程序
%x1，x2 分别为输入序列，n1，n2 为对应下标
%s=0 为加法；其他值为乘法
n=min(min(n1),min(n2)):max(max(n1),max(n2));
y1=zeros(1,length(n));
y2=y1;
y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;
if s==0
    y==y1+y2;
else
    y=y1.*y2;
end

function [yengy,ypower,yop]=sig_energy(x,n)
%计算信号 x 的能量和功率,并进行信号折叠
%n 为信号 x 的下标
yengy=sum(abs(x).^2);
ypower=yengy/length(x);
yop=fliplr(x);
n=-fliplr(n);

function [x_even,x_odd]=sig_evenodd(x,n)
%信号奇偶分解的程序
%x 为输入序列，n 为对应下标
m=fliplr(n);
mm=min([m,n]):max([m,n]);
nm=n(1)-m(1);
nl=1:length(n);
xl=zeros(1,length(mm));
xl(nl+nm)=x;
x=xl;
x_even=0.5*(x+fliplr(x));
x_odd=0.5*(x-fliplr(x));
```

【例 10.2】用 MATLAB 编写生成均值为 *a*，方差为 *b* 的高斯随机序列信号。

MATLAB 程序代码如下：

```
function y=randn_ab(N,a,b)
%产生均值为 a 方差为 b 的 N 点高斯随机序列
%N 为随机序列的长度
x=randn(1,N);
y=a+sqrt(b)*x;
plot(y);
grid;
```

以上是离散信号的产生例子，在实际操作中可以灵活运用 MATLAB 提供的各种运算和工具箱函数实现系统需要的信号形式。

分析一个离散系统最重要的是得到其系统单位抽样响应，用 filter 函数或者 impz 函数即可实现。

【例 10.3】求如图 10-9 所示的离散系统的单位抽样响应。

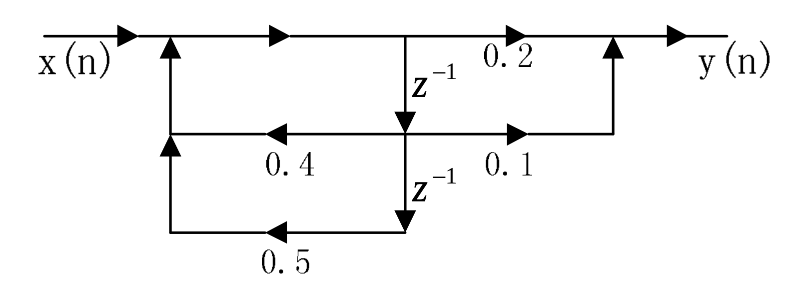


图 10-9　离散系统的信号流图

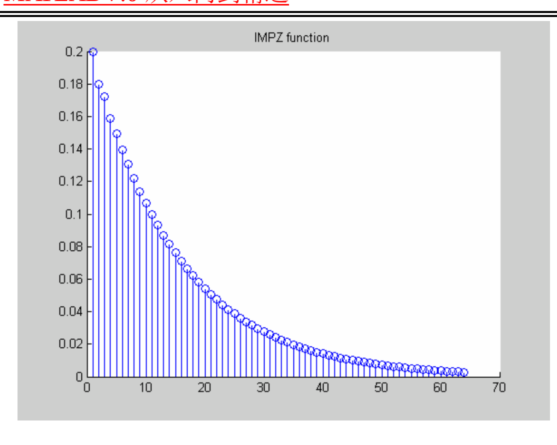
该离散系统对应的输入输出差分方程为：

$$y(n)-0.4y(n-1)-0.5y(n-2)=0.2x(n)+0.1x(n-1)$$

MATLAB 程序代码如下：

```
pul=[1 zeros(1,63)];%产生单位取样序列
b=[0.2 0.1];
a=[1 -0.4 -0.5];%b 和 a  都是系统参数
h=filter(b,a,pul);
hl=impz(b,a,64);
figure(1) %作图
stem(h)
title('FILTER function')
figure(2)
stem(hl)
title('IMPZ function')
```

程序采用 MATLAB 给出的两种函数 filter 和 impz 来计算，得到的系统单位响应曲线如图 10-10 所示，从图中可以看出两函数求得的结果相同。



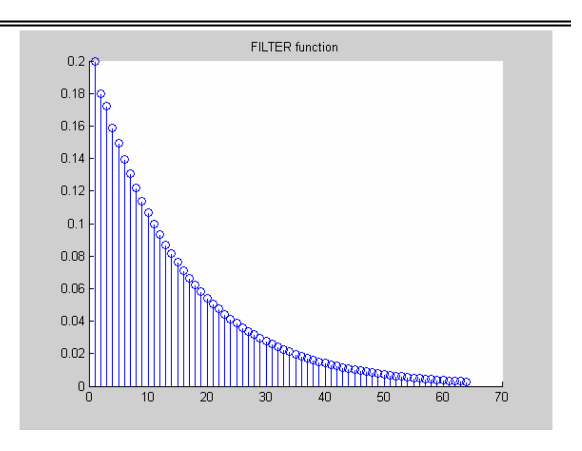


图 10-10　单位抽样响应

在进行离散系统的分析时，系统的频率响应和零极点增益也是常常需要考虑的问题，可分别用 freqz 函数和 roots 函数来实现。

10.3.2　离散傅立叶变换的 MATLAB 实现

周期序列实际上只有有限个序列值有意义，因此它的许多特性可以沿用到有限长序列上，由离散傅立叶级数的公式可以得到有限长序列的离散傅立叶变换的公式，在第 1 节中有具体的介绍。如果 $x(n)=\cos(\frac{np}{6})$ 是一个 N 为 12 的序列，利用 MATLAB 计算其 DFT 并画出图形。

【例 10.4】在 MATLAB 中 $x(n)=\cos(\frac{np}{6})$（N 为 12）DFT 的实现。

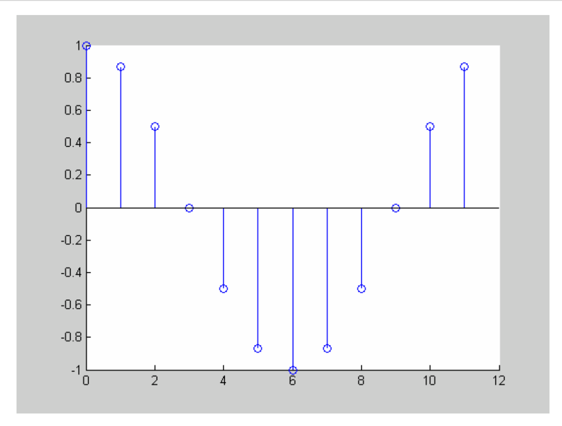
MATLAB 程序代码如下：

```
N=12;%序列长度
n=0:N-1;%时域取样
xn=cos(pi*n/6);%产生序列
k=0:N-1;%频域取样
wn=exp(-j*2*pi/N);
nk=n'*k;
wnnk=wn.^nk;
xk=xn*wnnk %计算 DFT
figure(1)%画图
stem(n,xn)
figure(2)
stem(k,abs(xk))
```

运行后得到的结果如下：

```
xk =
  Columns 1 through 7
```

```
        -0.0000    6.0000 + 0.0000i    -0.0000 - 0.0000i    -0.0000 - 0.0000i    -0.0000 - 0.0000i    -0.0000 - 0.0000i
-0.0000 - 0.0000i
        Columns 8 through 12
          0.0000 - 0.0000i    0.0000 - 0.0000i    0.0000 - 0.0000i    0.0000 - 0.0000i    6.0000 + 0.0000i
```



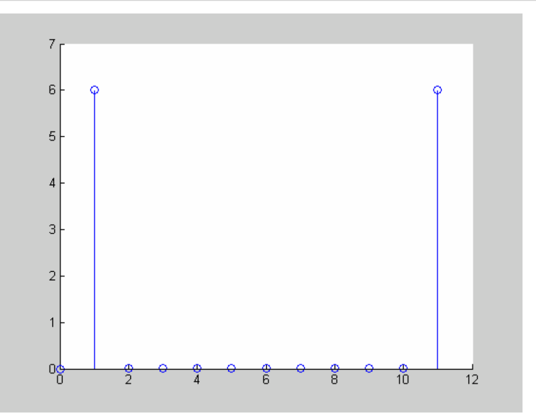


图 10-11 有限序列和序列的 DFT

同样由 IDFT 公式也可以编写相应的程序：

```
N=12;%序列长度
n=0:N-1;
k=0:N-1;
wn=exp(-j*2*pi/N);
nk=n'*k;
wnnk=wn.^(-nk);
xn=(xk*wnnk)/N;%计算 IDFT
figure(1)%画图
stem(k,abs(xk))
figure(2)
stem(n,real(xn))
```

在进行快速傅立叶变换 FFT 的操作时可以调用内部函数 fft，速度比较快。DFT 和 FFT 在信号处理中有着重要的应用，可以进行卷积运算，实现线性时不变系统等。

【例 10.5】设 $x(n)$ 长度为 N_1， $h(n)$ 长度为 N_2，计算两者的线性卷积，其步骤如下：

- 将 $x(n)$ 和 $h(n)$ 通过增加零值延长到 $N = N_1 + N_2 - 1$;
- 利用 FFT 计算 $x(n)$ 和 $h(n)$ 各自 N 点的 DFT;
- 计算 $Y(k) = X(k)H(k)$；
- 利用 IFFT 计算 $Y(k)$ 便可得到卷积结果 $y(n)$。

对于特定的序列，MATLAB 程序代码如下：

```
x=[2,3,1,4,5];
h=[2,1,7,4,5,7,2,3];
lenx=length(x);
lenh=length(h);
N=lenx+lenh-1;
xk=fft(x,N);
hk=fft(h,N);
yk=xk.*hk;
y=real(ifft(yk));
plot(y);
xlabel('n');
ylabel('y(n)');
title('x(n)*h(n)');
grid
```

运行结果如图 10-12 所示。

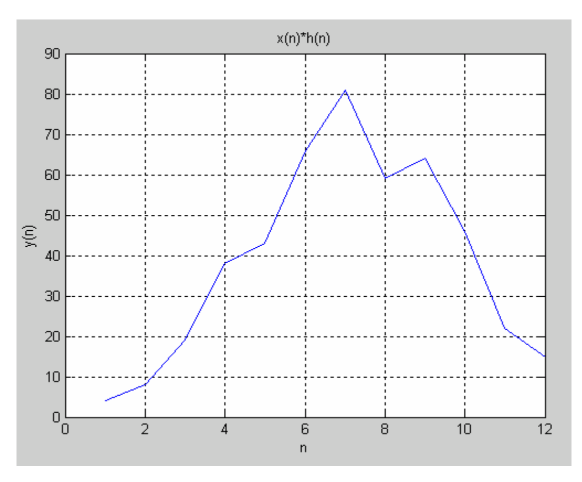


图 10-12　应用 FFT 计算线性卷积

10.3.3　Z 变换的 MATLAB 实现

有关 Z 变换的定义和性质在 10.1 节中已有介绍，对于 Z 变换为 $X(z)$ 的序列，MATLAB 的表示是通过 $X(z)$ 的系数实现的，Z 变换经常用到 deconv、residuez 和 freqz 等函数。

【例 10.6】计算 $X(z)=\dfrac{1}{(1-0.9z^{-1})^2(1+0.9z^{-1})}$　$|z|>0.9$ 的逆 Z 变换。

其 MATLAB 程序代码如下：

```
b=1;
a=poly([0.9 0.9 -0.9]);
[R,P,C]=residuez(b,a)
```

执行完毕的结果代码为：

```
R =
    0.2500
    0.5000
    0.2500
P =
    0.9000
    0.9000
   -0.9000
C =
     []
```

因此得到下面的表达式：

$$X(z)=\frac{0.25}{1-0.9z^{-1}}+\frac{0.5}{(1-0.9z^{-1})^{2}}+\frac{0.25}{1+0.9z^{-1}}\qquad |z|>0.9$$

相应的逆 Z 变换为下面的表达式：

$$x(n)=0.25(0.9)^{n}u(n)+\frac{5}{9}(n+1)(0.9)^{n+1}u(n+1)+0.25(-0.9)^{n}u(n)$$

$$=0.75(0.9)^{n}u(n)+0.5n(0.9)^{n+1}u(n)+0.25(-0.9)^{n}u(n)$$

【例 10.7】用 MATLAB 实现由系统的差分方程求出系统的冲激响应，画出其幅度相位响应，并指出系统的零、极点，差分方程为：

$$y(n)=y(n-1)-0.6y(n-2)+x(n)-2x(n-1)$$

MATLAB 程序代码如下：

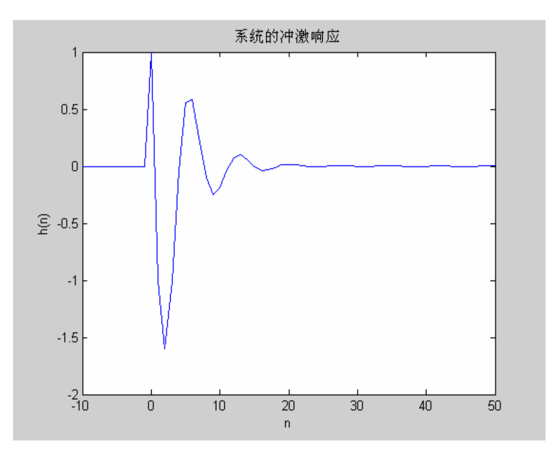
```
%由系统的差分方程求出系统的冲激响应，画出其幅度、相位响应，并指出系统的零极点
A=[1,-1,0.6];
B=[1,-2];
%指出系统的零极点
[R,p,C]=residuez(B,A)
%求系统的冲激响应
n=[-10:50];
x=[(n)==0];
h=filter(B,A,x);
figure(1);
plot(n,h);
xlabel('n');
```

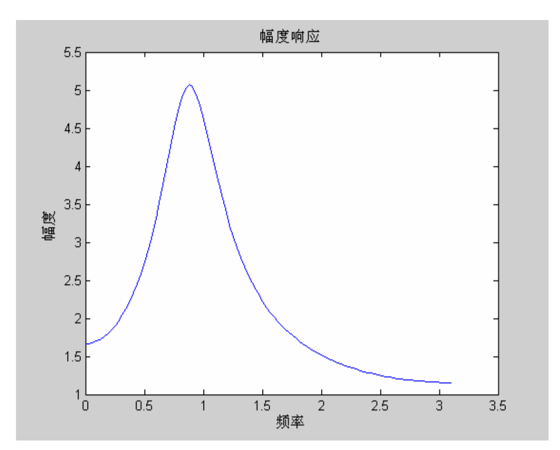
```
ylabel('h(n)');
title('系统的冲激响应');
%求其幅度相位响应
[H,W]=freqz(B,A,100);
magH=abs(H);
phaH=angle(H);
figure(2);
plot(W,magH);
xlabel('频率');
ylabel('幅度');
title('幅度响应');
figure(3);
plot(W,phaH);
xlabel('频率');
ylabel('幅度');
title('相位响应');
```

可得系统的零极点为：

```
R =
    0.5000 + 1.2677i
    0.5000 - 1.2677i
p =
    0.5000 + 0.5916i
    0.5000 - 0.5916i
C =
       []
```

系统的冲激响应、幅度响应和相位响应如图 10-13 所示。





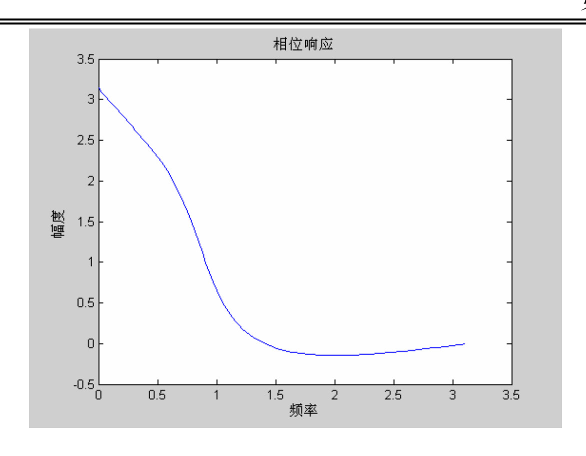


图 10-13　系统的冲激响应、幅度响应和相位响应

此外，Z 变换还可用于差分方程的求解。

10.3.4　FIR 滤波器的 MATLAB 实现

基于 MATLAB 的 FIR 滤波器的设计有多种方法，包括窗函数法（对应的 MATLAB 函数有 fir1、fir2、kaiserord）、最优化设计法（对应的 MATLAB 函数有 firls、remez、remezord）、最小二乘约束设计法（fircls、fircls1）、非线性相位滤波器设计法（cremez）和升余弦方法（firrcos）本节将举例说明 fir1、fir2、kaiserord 和 firls、remez 如何用于滤波器的设计。

【例 10.8】设计一个阶数为 48，通带范围为 0.35≤ w ≤0.65 的带通 FIR 线性相位滤波器，并分析它的频率特性。

MATLAB 程序代码如下：

```
b=fir1(48,[0.35 0.65]);
freqz(b);
```

得到的频率特性如图 10-14 所示。

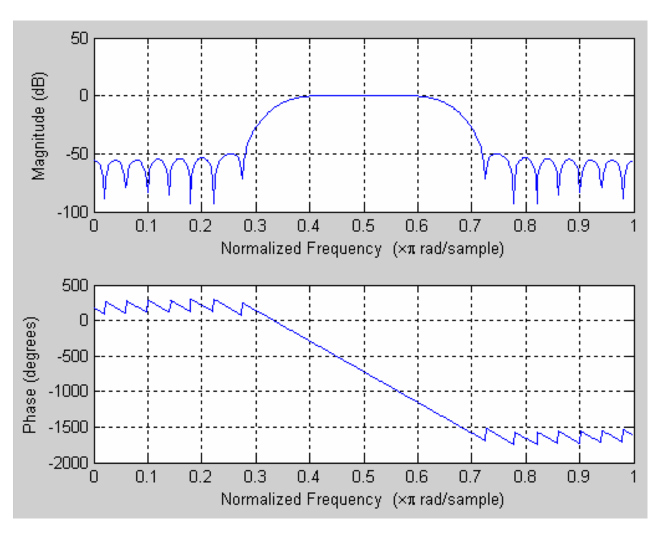


图 10-14　利用 fir1 设计的带通滤波器的频率响应

【例 10.9】设计一个 60 阶的滤波器，要求设计的滤波器 0～$p/8$ 的幅度为 1，在 $p/8$～$2p/8$ 的幅度为 1/2，在 $2p/8$～$4p/8$ 的幅度响应为 1/4，在 $4p/8$～$6p/8$ 的幅度响应为 1/6，在 $6p/8$～p 的响应为 1/8，并且画出理想滤波器和设计得到的滤波器的幅度频率响应进行比较。

MATLAB 程序代码如下：

```
f=[0 0.125 0.125 0.250 0.250 0.500 0.500 0.750 0.750 1.00];
m=[1 1 0.5 0.5 0.25 0.25 1/6 1/6 0.125 0.125];
b=fir2(60,f,m);
[h,w]=freqz(b);
plot(f,m,w/pi,abs(h))
grid on;
title('设计滤波器和理想滤波器幅度频率特性比较');
xlabel('归一化频率(xfs)');
ylabel('幅度');
```

得到的频率响应如图 10-15 所示。

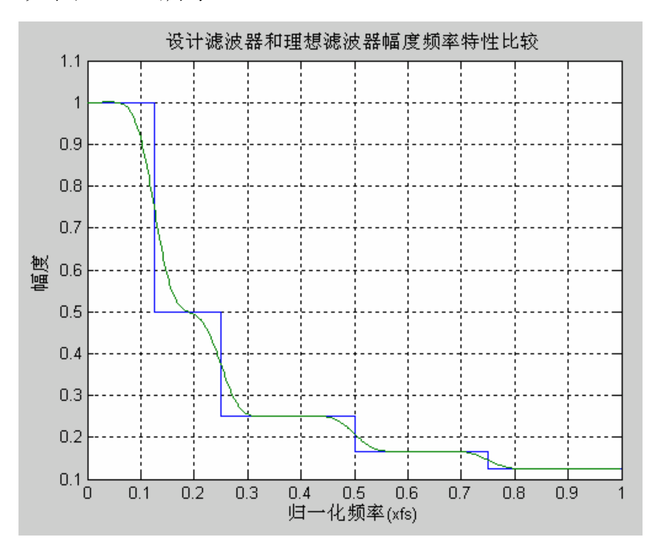


图 10-15　利用 fir2 设计的任意频率响应的滤波器

【例 10.10】利用凯塞窗函数设计一个长度为奇数的带通滤波器（长度为奇数对应滤波器是偶阶的，由此能满足带通滤波器的要求），通带范围是 1300Hz～2210Hz，阻带范围是 0Hz～1000Hz、2410Hz～4000Hz，阻带的波纹最大为 0.01，通带的波纹最大为 0.05，信号的采样频率为 8000Hz。

MATLAB 程序代码如下：

```
fsamp=8000;
fcuts=[1000 1300 2210 2410];
mags=[0 1 0];
devs=[0.01 0.05 0.01];
[n,wn,beta,ftype]=kaiserord(fcuts,mags,devs,fsamp);
```

```
n=n+rem(n,2);
hh=fir1(n,wn,ftype,kaiser(n+1,beta),'noscale');
[H,f]=freqz(hh,1,1024,fsamp);
plot(f,abs(H));
grid on
```

得到的滤波器的阶数为 90，滤波器的截止频率分别为 0.2875 和 0.5775，凯塞窗函数的 b 值为 3.3953。滤波器的频率响应如图 10-16 所示。

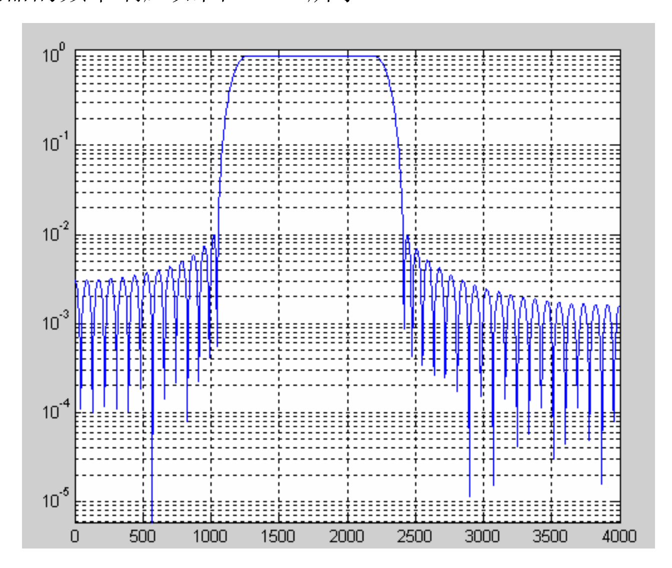


图 10-16　利用 kaiserord 设计的滤波器的频率响应

【例 10.11】分别用函数 firls 和 remez 设计一个 30 阶的 FIR 低通滤波器，通带边界频率为 $0.4*\pi$，幅值为 3，阻带边界频率为 $0.5*\pi$，幅值为 1，并比较两函数设计的 FIR 滤波器。

> 注意：函数 firls 是 fir1 和 fir2 的扩展，其基本设计准则是利用最小二乘法使期望的频率响应和实际的频率响应之间的整体误差最小。函数 remez 实现 Parks-Mcclellan 算法，它利用 Remez 交换法和 Chebyshev 近似理论来设计滤波器，使实际频率响应拟合频率响应达到最优。

MATLAB 程序代码如下：

```
n=30;
f=[0 0.4 0.5 1];
a=[1 1 0 0];
b=firls(n,f,a);
[h,w]=freqz(b);
bb=remez(n,f,a);
[hh,w]=freqz(bb);
axes('position',[0.2 0.2 0.5 0.5]);
plot(w/pi,abs(h),'b-',w/pi,abs(hh),'r--');
xlabel('frequency(pi)');
```

```
ylabel('magnitude');
title('direct IIR design-yulewalk');
legend('firls','remez');
grid
```

程序运行结果如图 10-17 所示。

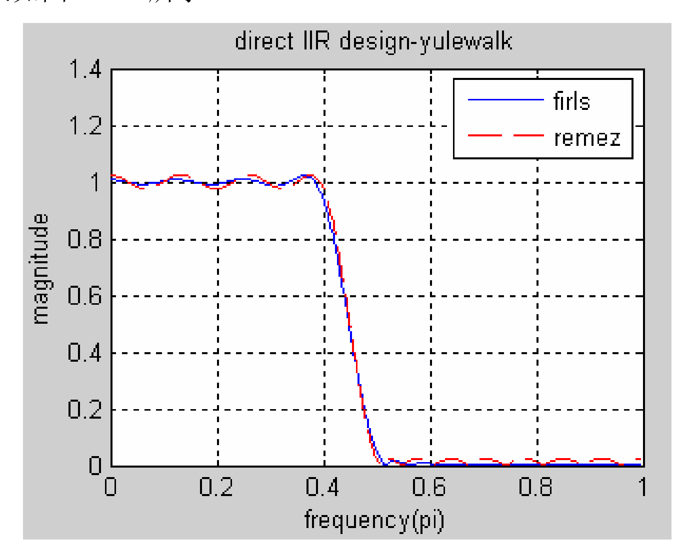


图 10-17　FIR 滤波器频率响应

可见用 firls 设计的滤波器在整个频率范围内均具有良好的频率响应，但理想频率响应和实际频率响应的误差在带区内不均匀，且在边界频率处误差较大，而用函数 remez 设计的滤波器在通带内有等波纹特性，在边界频率 $0.4*\pi$ 和 $0.5*\pi$ 处及过渡带内更接近于理想频响。

10.3.5　IIR 滤波器的 MATLAB 实现

IIR 滤波器的设计主要有模拟滤波器变换法（经典设计法）、直接设计法和最大平滑滤波器设计法三种方法。本节将列举具有代表性的例程来说明三种方法的应用。

- 经典设计法是基于模拟滤波器的变换原理，首先根据滤波器的技术指标设计出相应的模拟滤波器，然后再离散化为满足给定技术指标的数字滤波器。在 MATLAB 中对应的工具函数有完全设计函数——butter、cheby1、cheby2、ellip、besself；阶数估计函数——buttord、cheb1ord、cheb2ord、ellipord；低通模拟原型滤波器函数——buttap、cheb1ap、cheb2ap、ellipap；频率转换函数——lp2lp、lp2bp、lp2hp、lp2bs；滤波器离散化函数——bilinear、impinvar。
- 直接设计法是在离散域内用最小二乘法逼近给定的幅频特性，在 MATLAB 中对应的工具函数是 yulewalk。
- 最大平滑滤波器设计法是设计一般化低通滤波器，其零点数多于极点，在 MATLAB 中对应的工具函数是 maxflat。

【例 10.12】经典法设计滤波器的设计有脉冲响应不变法和双线性变换法两种方式。

- 用椭圆滤波器原型设计一个低通滤波器，满足 $w_p = 0.2*\pi$、$R_p = 0.5\text{dB}$、$w_s = 0.3*\pi$、

$A_s = 20\text{dB}$，采用脉冲响应不变法。

MATLAB 程序代码如下：

```
wp=0.2*pi;
ws=0.3*pi;
rp=0.5;
rs=20;
[n,wn]=ellipord(wp,ws,rp,rs,'s')
[z,p,k] = ellipap(n,rp,rs);
w = logspace(-1,1,1000);
h = freqs(k*poly(z),poly(p),w);
semilogx(w,abs(h));
grid
```

运行结果如图 10-18 所示。

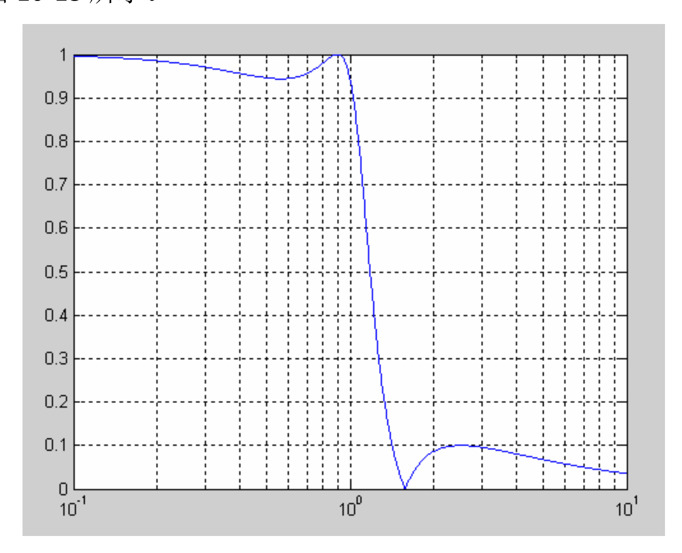


图 10-18　椭圆低通滤波器的频响特性

- 采用双线性变换法设计带通 Chebyshev Ⅰ型数字滤波器，要求通带边界频率为 100～200Hz; 通带纹波小于 3dB; 阻带衰减大于 30dB; 过渡带宽为 30Hz; 采样频率为 1000Hz。

MATLAB 程序代码如下：

```
fs=1000;
wp=[100 200]*2/fs;
ws=[30 300]*2/fs;
rp=3;
rs=30;
Nn=128;
[N,wn]=cheb1ord(wp,ws,rp,rs)
[b,a]=cheby1(N,rp,wn)
```

```
freqz(b,a,Nn,fs)
```

程序运行结果如图 10-19 所示。

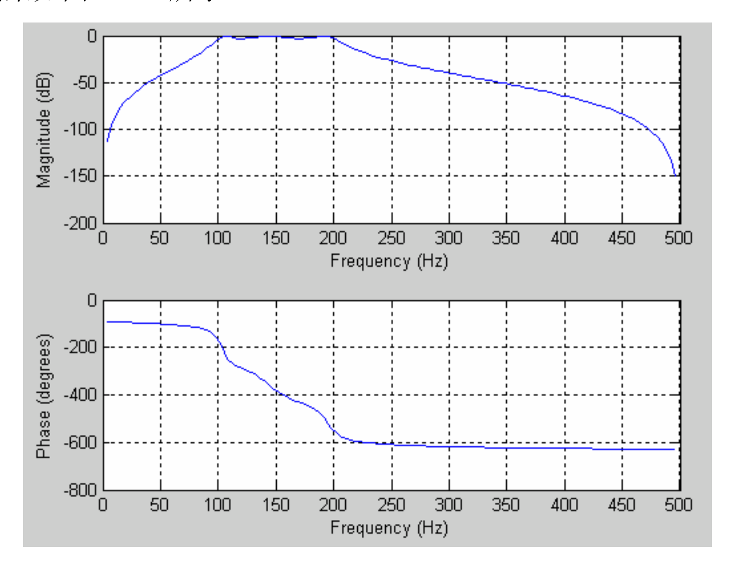


图 10-19　Chebyshev Ⅰ型数字滤波器的频率特性图

【例 10.13】用直接法设计一个多频带数字滤波器，幅频响应值如下：

f=[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1];

m=[0 0 1 1 0 0 1 1 0 0 0];

采用函数 yulewalk 来实现，具体操作步骤如下：

- 计算与分子多项式相应的幅值平方响应的辅助分子式和分母式;
- 由辅助分子式和分母式计算完全的频率响应;
- 计算滤波器的脉冲响应;
- 采用最小二乘法拟合脉冲响应，最终求得滤波器的分子多项式。

MATLAB 程序代码如下：

```
oder=10;
f=0:0.1:1;
m=[0 0 1 1 0 0 1 1 1 0 0];
[b,a]=yulewalk(oder,f,m)
[h,w]=freqz(b,a,128)
axes('position',[0.2 0.2 0.4 0.4]);
plot(f,m,'b-',w/pi,abs(h),'m--');
xlabel('frequency(pi)');
ylabel('magnitude');
title('direct IIR design-yulewalk');
legend('理想图形','实际图形',1);
grid
```

程序运行结果如图 10-20 所示。

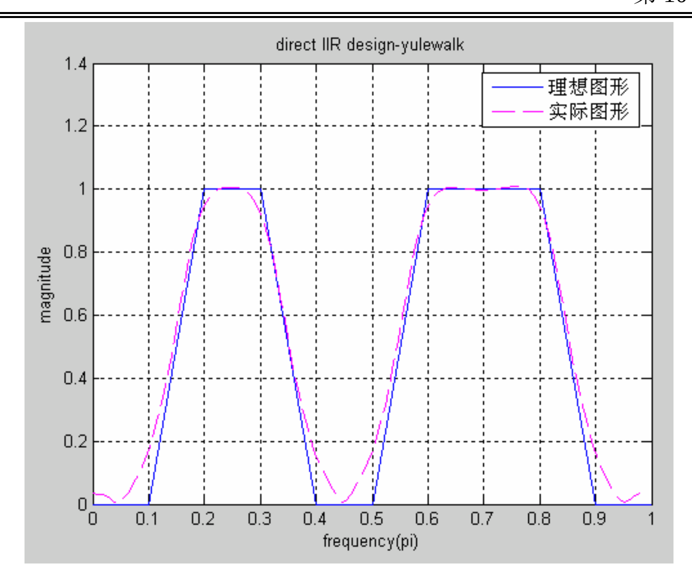


图 10-20　直接设计的多频带滤波器

【例 10.14】用 maxflat 函数设计一个通用 Butter-Worth 低通滤波器，满足系统函数分子阶数为 8 阶，系统函数分母阶数为 3 阶，截止频率为1*π。

MATLAB 程序代码如下：

```
nb=8;
na=3;
wn=0.6;
[b,a]=maxflat(nb,na,wn,'plots')
maxflat(nb,na,wn,'trace')
```

程序运行结果如下：

```
b =
    0.1650    0.5048    0.4100    -0.1134    -0.2329    -0.0244    0.0202    -0.0043    0.0004
a =
    1.0000    -0.1813    0.2073    -0.3006
 Table:
    L         M         N         wo_min/pi    wo_max/pi
    8.0000    0         3.0000    0            0.2919
    7.0000    1.0000    3.0000    0.2919       0.4021
    6.0000    2.0000    3.0000    0.4021       0.5000
    5.0000    3.0000    3.0000    0.5000       0.5979
    4.0000    4.0000    3.0000    0.5979       0.7081
    3.0000    5.0000    3.0000    0.7081       1.0000
ans =
    0.1650    0.5048    0.4100    -0.1134    -0.2329    -0.0244    0.0202    -0.0043    0.0004
```

由上述代码得到如图 10-21 所示的图形。

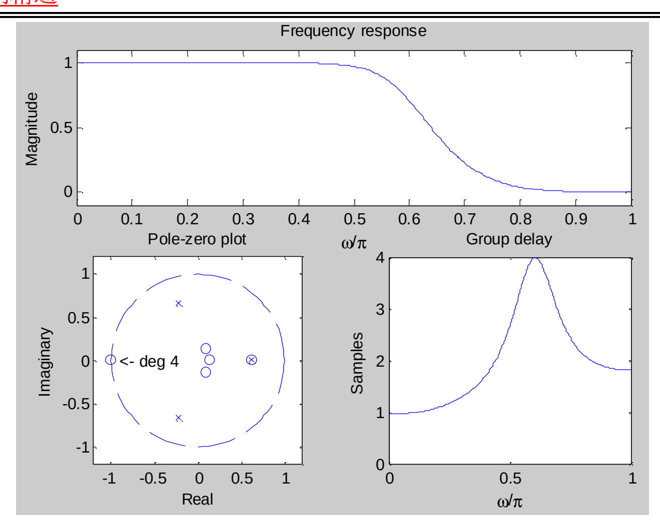


图 10-21　最大平滑 Butter-Worth 低通滤波器特性

第 11 章　图像处理工具箱

在上一章学习了 MATLAB 信号处理方面的应用之后，本章将在对各种图像处理方法的理论作系统讲解的基础上，详细介绍了 MATLAB 图像处理工具箱函数的使用方法，并给出了大量的应用实例，使读者能够更好地掌握和使用 MATLAB 图像处理工具箱函数进行图像处理。

11.1　图像处理工具箱介绍

11.1.1　常用图像格式

图像格式指的是存储图像采用的格式，不同的操作系统、不同的图像处理软件，所支持的图像格式都有可能不同。在实际应用中经常会遇到的图像格式有 BMP、GIF、TIFF、PCX、JPEG、PSD、PCD 和 WMF 等。由于篇幅的限制，本节仅介绍其中的一部分。

1．BMP 文件

BMP 文件是 Microsoft Windows 所定义的图像文件格式，最早应用在微软公司的 Microsoft Windows 窗口系统中。BMP 图像文件具有以下特点：

- 只存放一幅图像；
- 只能存储单色、16 色、256 色和真彩色 4 种图像数据；
- 图像数据有压缩和非压缩两种处理方式；
- 调色板的数据存储关系较为特殊，存储格式不是固定的，而是与文件头的某些具体参数（如像素位 bbp、压缩算法等）密切相关的。其中，Windows 设有 RLE4 和 RLE8 两种压缩方式，RLE4 只能处理 16 色图像数据，RLE8 则只能压缩 256 色图像数据。

BMP 图像文件的文件结构可分为表头、调色板和图像数据三部分。其中表头长度固定为 54 个字节，只有真彩色 BMP 图像文件内没有调色板数据，其余不超过 256 种颜色的图像文件都必须有调色板信息。

BMP 文件头数据结构含有 BMP 文件的类型、大小和打印格式等信息。在 Windows 种对其进行了定义，其定义如下：

```
Typedef struct tag BITMAPFILEHEADER{
WORD      bftype;              /*位图文件的类型，必须为 BMP
DWORD     bfSize;              /*位图文件的大小，以字节为单位
WORD      bfReserved1;         /*位图文件保留字，必须为 0
WORD      bfReserved2;         /*位图文件保留字，必须为 0
```

```
DWORD    bfoffBits;              /*位图阵列的起始位置，以相对于位图文件头的偏移量表示
}BITMAPFILEHEADER;
```

可见，位图信息数据结构含有位图文件的尺寸和颜色等信息。位图阵列记录了位图的每一个像素值。在生成位图文件时，Windows 从位图的左下脚开始逐行扫描位图，将位图的像素值一一记录下来，组成了位图阵列。

2．GIF 文件

GIF（Graphics Interchange Format）图像文件格式是 CompuServe 公司最先在网络中用于在线传输图像数据。GIF 图像文件经常用于网页的动画、透明等特技制作。该文件具有以下特点：

- 文件具有多元化结构，能够存储多张图像，并可以进行多图像的定序或覆盖，交错屏幕绘图以及文本覆盖等功能。
- 调色板数据有通用调色板和局部调色板之分。
- 采用了 LZW 压缩法。
- 图像数据一个字节存储一个像素点。
- 文件内的各种图像数据区和补充区多数没有固定的数据长度和存储位置，为了方便程序寻找数据区，就以数据区的第一个字节作为标识符，以使程序能够判断读到哪种数据区。
- 图像数据有顺序排列和交叉排列两种方式。
- 图像最多只能存储 256 色图像。

GIF 图像文件结构一般由表头、通用调色板、图像数据区以及 4 个补充区这 7 个数据单元组成。其中，表头和图像数据区是不可缺少的单元，通用调色板和其余的 4 个补充区是可选择的内容。

3．TIF 文件

TIF（Tag Image File Format）图像文件格式是现有图像文件格式中最复杂的一种，它是由 Aldus 公司与微软公司开发设计的图像文件格式，提供了各种信息存储的完备手段。其主要特点如下：

- 应用指针功能，实现多幅图像存储；
- 文件内数据没有固定的排列顺序，但规定表头必须在文件前端，标识信息区和图像数据区在文件中可以任意存放；
- 可定制私人用的标识信息；
- 能够接受除了一般图像处理 RGB 模式之外的 CMYK、YcbCr 等多种不同的图像模式；
- 可存储多份调色板数据，其调色板的数据类型和排列顺序较为特殊；
- 能够提供多种不同的压缩数据的方法；
- 图像数据可分割成几个部分进行分别存档。

TIF 图像文件主要由表头、标识信息区和图像数据区 3 个部分组成。其中，文件内固定只有一个位于文件前端表头，表头由一个标志参数指出标识信息区在文件中的存储地址，标识信息区有多组用于存储图像数据区的地址。每组标识信息长度固定为 12 个字节，前 8 个字节分别代表信息的代号（2 个字节）、数据类型（2 个字节）、数据量（4 个字节），最后 4 个

字节用于存储数据值或标识参数。

4．JPEG 格式

JPEG（Joint Photographic Experts Group）是对精致灰度或彩色图像的一种国际压缩标准，其全称为"连续色调静态图像的数字压缩和编码"，已在数字照相机上得到了广泛应用，当选用有损压缩方式时可以节省相当大的空间。

JPEG 标准只是定义了一个规范的编码数据流，并没有规定图像数据文件的格式。Cube Microsystems 公司定义了一种 JPEG 文件交换格式（JFIF-File Interchange Format）。JFIF 图像是一种使用灰度标识，或者使用 Y，C_b，C_r 分量彩色表示的 JPEG 图像。它包含一个与 JPEG 兼容的头。一个 JFIF 文件通常包含单个图像，图像可以是灰度的（其中的数据为单个分量），也可以是彩色的。

11.1.2　MATLAB 7.0 图像类型

图像类型是指数组数值与像素颜色之间定义的关系，它与图像格式概念有所不同，在 MATLAB 7.0 图像处理工具箱中，有 5 种类型的图像，下面分别介绍其基本情况。

1．二进制图像

在一幅二进制图像中，每一个像素将取两个离散数值（0 或 1）中的一个，从本质上说，这两个数值分别代表状态"开"或"关"。

二进制图像使用 unit8 或双精度类型的数组类存储。由于 unit8 数组使用的内存较小，故 unit8 类型的数组通常比双精度类型的数组性能要好。在图像处理工具箱中，任何一个返回一幅二进制图像的函数均使用 unit8 逻辑数组存储该图像，并且使用一个逻辑标志来标识 unit8 逻辑数组的数据范围。若逻辑状态为"开"，则数组范围为[0,1]；若为"关"，则数组范围为[0,255]。图 11-1 是一幅典型的二进制图像实例。

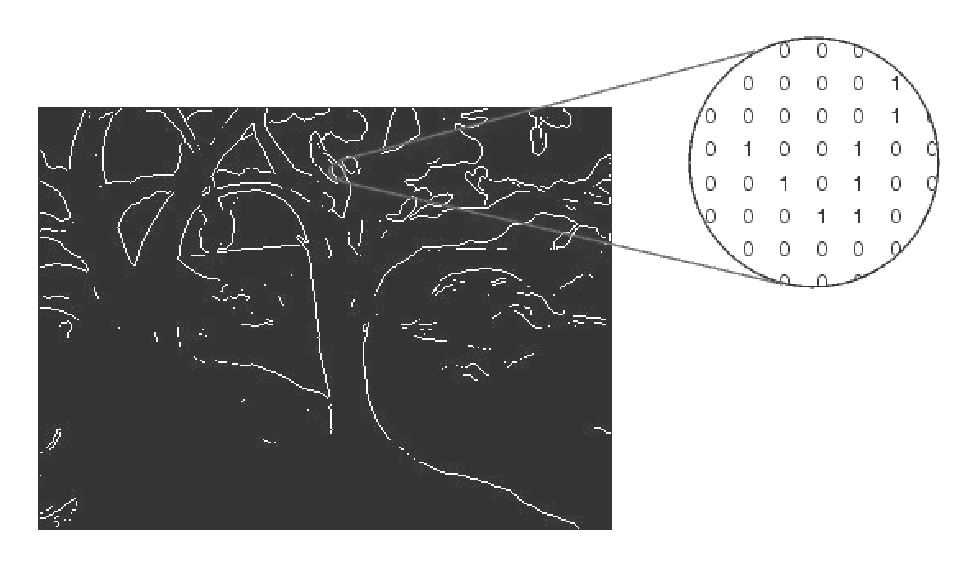


图 11-1　典型的二进制图像实例

2．索引图像

索引图像是一种把像素值直接作为 RGB 调色板下标的图像。在 MATLAB 7.0 中，索引图像包含一个数据矩阵 *X* 和一个颜色映射矩阵 map。其中，数据矩阵可以是 unit8、unit16 或双精度类型的，颜色映射矩阵 map 是一个 m×3 的数据帧里，其中每个元素的值均为[0,1]之间的双精度浮点类型数据，map 矩阵的每一行分别标识红色、绿色和蓝色的颜色值。索引图像可把像素值直接映射为调色板数据，每一个像素的颜色通过使用 X 的数值作为 map 的下标来获得，如值 1 指向 map 中的第一行，值 2 指向第二行，依次类推。

颜色映射通常与索引图像存储在一起，当装载图像时，MATLAB 7.0 自动将颜色映射表与图像同时装载。图 11-2 显示了索引图像的结构。该图像中的像素用整数类型标识，这个整数将作为存储在颜色映射表中的颜色数据的指针。

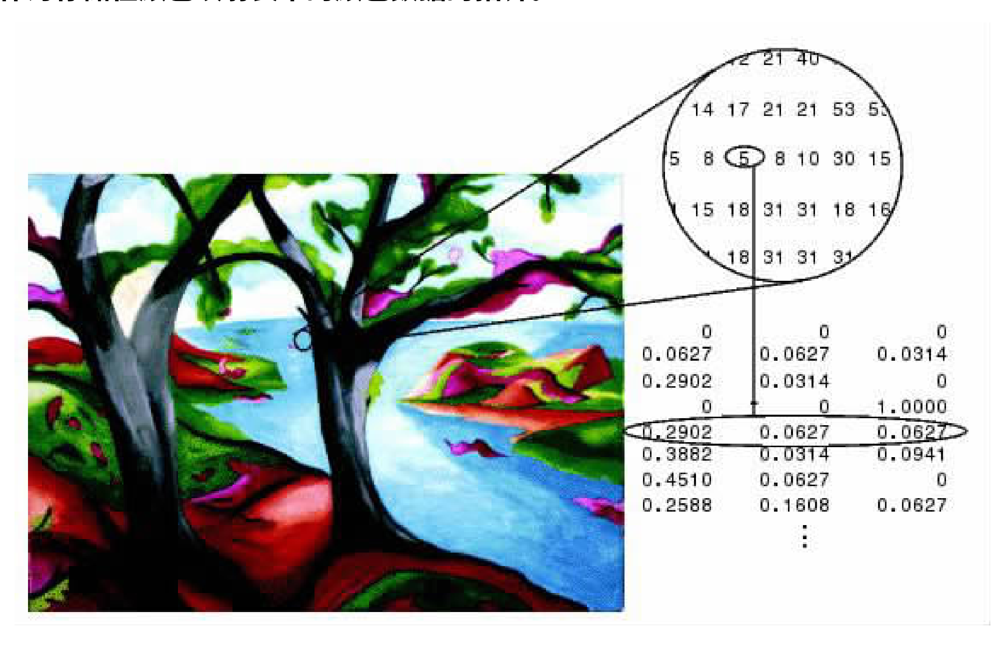


图 11-2　索引图像实例

图像矩阵与颜色映射表之间的关系依赖于图像数据矩阵的类型。如果图像数据矩阵是双精度类型，则数据 1 指向 map 矩阵中的第一行，数据值 2 指向 map 中的第二行，依次类推；如果图像矩阵是 unit8 或 unit16 类型时，将产生一个偏移量，即数据 0 标识矩阵 map 中的第一行，数据 1 将指向 map 中的第二行，依次类推。在如图 11-2 所示的图像中，图像矩阵用的是双精度类型，无偏移量，数值 5 指向颜色映射表中的第五行。

3．灰度图像

灰度图像通常由一个 unit8、unit16 或双精度类型的数组来描述，其实质是一个数据矩阵 *I*，该矩阵中的数据均代表了一定范围内的灰度级，每一个元素与图像的一个像素点相对应，通常 0 代表黑色，1、255 或 65635（针对不同存储类型）代表白色。大多数情况下，灰度图像很少和颜色映射表一起保存，但是在显示灰度图像时，MATLAB 7.0 仍然在后台使用预定义的默认灰度颜色映射表。图 11-3 为一个典型的双精度灰度图像。

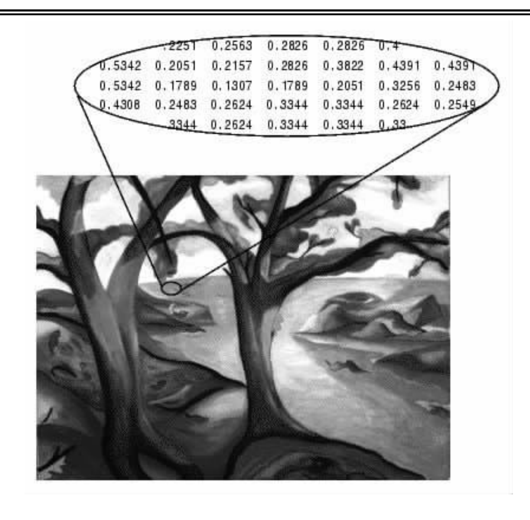


图 11-3　典型的双精度灰度图像

4．多帧图像

多帧图像是一种包含多幅图像或帧的图像文件，又称为多页图像或图像序列，它主要用于需要对时间或场景集合进行操作的场合，例如磁谐振图像切片或电影帧等。在 MATLAB 7.0 中，它是一个四维数组，其中第四维用来指定帧的序号。

在 MATLAB 7.0 图像处理工具箱中提供了同一个数组中存储多幅图像的支持，每一幅单独的图像成为一帧。如果一个数组中包含多帧，那么这些图像在四维中是相联系的。在一个多帧图像数组中，每一幅图像必须有相同的大小和颜色分量。在多帧图像中，每一幅图像还要使用相同的调色板。另外，图像处理工具箱中的许多函数（如 imshow）只能对多帧图像矩阵的前两维或三维进行操作，也可以对四维数组使用这些函数，但是必须单独处理每一帧。如果将一个数组传递给一个函数，并且数组的维数超过该函数设计的操作位数，那么得到的结果是不可预知的。

5．RGB 图像

RGB 图像又称为真彩图像，它是利用 R、G、B 三个分量标识一个像素的颜色，R、G、B 分别代表红、绿、蓝 3 种不同的颜色，通过三基色可以合成出任意颜色。所以对一个尺寸维 $n\times m$ 的彩色图像来说，在 MATLAB 7.0 中则存储一个 $n\times m\times 3$ 的多维数据数组，其中数组中的元素定义了图像中每一个像素的红、绿。蓝颜色值。值得注意的是 RGB 图像不使用调色板，每一个像素的颜色由存储在相应位置的红、绿、蓝颜色分量的组合来确定，图形文件格式把 RGB 图像存储维 24 位的图像，红、绿、蓝分量分别占用 8 位，因而图像理论上可以有 2^{24}＝16 777 216 种颜色，由于这种颜色精度能够再现图像有真实色彩，故称 RGB 图像为真彩图像。

图 11-4 为一幅典型的双精度 RGB 图像，在此图种，为了确定像素（2，3）的颜色，需要查看一组数据 RGB（2，3，1∶3）。假设（2，3，1）数据为 0.5176，（2，3，2）数值为 0.1608，（2，3，3）数值为 0.0627，则像素（2，3）的 RGB 颜色为（0.5176 红色，0.1608 绿色，0.0627 蓝色）。

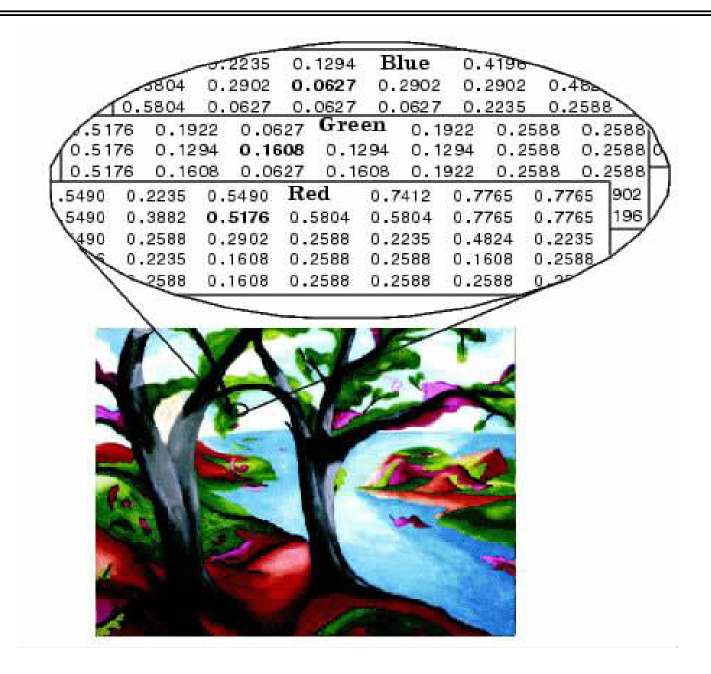


图 11-4　典型的双精度 RGB 图像实例

11.1.3　MATLAB 7.0 图像类型转换

在有些图像操作中，需要对图像的类型进行转换。比如要对一副索引图像的彩色图像进行滤波，首先应该将其转换成 RGB 图像，此时在对 RGB 图像使用滤波器时，MATLAB 7.0 将恰当地滤掉图像中的部分灰度值。如果不将索引图像进行转换，MATLAB 7.0 则对图像调色板的序号进行滤波，这样得到的结果将没有任何意义。下面对一些 MATLAB 7.0 图像处理工具箱中常用的类型转换进行介绍。

1．图像颜色浓淡处理（图像抖动）

dither 函数通过抖动算法转换图像类型，其语法格式为：

- X = dither(RGB,map)：通过抖动算法将真彩色图像 RGB 按指定的颜色图（调色板）*map* 转换成索引色图像 *X*。
- X = dither(RGB,map,Qm,Qe)：利用给定的参数 Qm，Qe 从真彩色图像 RGB 种产生索引色图像 *X*。Qm 表示沿每个颜色轴反转颜色图的量化（即对于补色各颜色轴）的位数，Qe 表示颜色空间计算误差的量化位数。如果 Qe<Qm，则不进行抖动操作。

2．灰度图像转换为索引图像

gray2ind 函数可以将灰度图像转换成索引图像，其语法格式如下：

- [X,map] = gray2ind(I,n)：按指定的灰度级数 *n* 和颜色图 *map*，将灰度图像 *I* 转换成索引色图像 *X*，*n* 的默认值为 64。

3．索引图像转换为灰度图像

ind2gray 函数可以将索引图像转换为灰度图像，其语法格式如下：

- I = ind2gray(X,map)：将具有颜色图 *map* 的索引色图像 *I* 转换成灰度图像 *X*，去掉了图像的色度和饱和度，仅保留了图像的亮度信息。输入图像可以是 double 或 unit8 类型，输出图像为 double 类型。

4．RGB 图像转换为灰度图像

rgb2gray 函数用于将一幅真彩图像（RGB）转换成灰度图像，其语法格式如下：

- I = rgb2gray(RGB)：将真彩色图像 RGB 转换成灰度图像 I；
- newmap = rgb2gray(map)：将颜色图 *map* 转换成灰度图像 I。

注意：如果输入的是真彩色图像，则图像可以是无符号类型或双精度类型，输出图像 *I* 与输入图像类型相同。如果输入的是颜色图，则输入和输出的图像均为双精度类型。

5．RGB 图像转换为索引图像

rgb2ind 函数用于将真彩图像转换成为索引图像，可采用直接转换、均匀量化、最小量化、颜色图近似 4 种方法。除直接转换方法外，其他方法在不指定选项 nodither 时自动进行图像抖动。其语法格式为：

- [X,map] = rgb2ind(RGB)：直接将 RGB 图像转化为具有颜色图 *map* 的矩阵 ***X***。由于每个像素点具有一个值，转换后的颜色图可能很长；
- [X,map] = rgb2ind(RGB,tol)：用均匀量化的方法将 RGB 图像转换为索引图像 ***X***。*map* 包括至少（floor(1/tol)+1）^{3} 个颜色，*tol* 的范围从 0.0～1.0；
- [X,map] = rgb2ind(RGB,n)：使用最小方差量化方法将 RGB 图像转换为索引图像，*map* 中包括至少 *n* 个颜色；
- X = rgb2ind(RGB,map)：通过将 RGB 中的颜色与颜色图 *map* 中最相近的颜色匹配，将 RGB 转换为具有 *map* 颜色图的索引图像。

6．索引图像转换为 RGB 图像

ind2rgb 函数将索引图像转换成真彩图像，其语法格式为：

- RGB = ind2rgb(X,map)：将矩阵 ***X*** 及相应的颜色图 *map* 转换成真彩图像 RGB。实际实现时，就是产生一个三维数组，然后将索引图像的颜色图的颜色赋予三维数组。输入图像 *X* 可以是双精度类型或 8 位无符号类型，输出图像 RGB 为双精度类型。

7．通过阈值化方法将图像转换为二值图像

im2bw 函数通过设置亮度阈值将真彩图像、索引图像以及灰度图像转换成二值图像。在转换过程中，如果输入图像不是灰度图像，首先将其转换为灰度级图像，然后通过阈值化将灰度级图像转换成二值图像。输出二值图像和输入图像之间的关系是在输入图像所有亮度小于给定值（*level* 取值范围为[0,1]）的像素点处均为 0，其他均为 1。其语法格式为：

- BW = im2bw(I,level)：输入二值图像为 *I*，*level* 为给定的阈值。

8．通过阈值化方法从灰度图像产生索引图像

grayslice 函数通过设定阈值将灰度图像转换成索引图像，其语法格式为：

- X = grayslice(I,n)：将灰度图像 *I* 均匀量化为 *n* 个等级，然后转换为伪彩色图像 *X*；
- X = grayslice(I,v)：按指定的阈值向量 *v*（每一个元素都在 0～1 之间）对图像 *I* 的值域进行划分，然后转换成索引图像 *X*。

值得注意的是输入图像 *I* 可以是双精度类型或 8 位无符号类型。如果阈值数量小于 256，则返回图像 *X* 的数据类型是 8 位无符号类型，*X* 的值域为[0,n]或[0,length(v)]；否则，返回图像 *X* 位双精度类型，值域为[0,n+1]或[1,length(v)+1]。

9．将矩阵转换为灰度图像

mat2gray 函数用于将一个数据矩阵转换为一幅灰度图像，其语法格式为：

- I = mat2gray(X,[xmin xmax])：按指定的取值区间[*xmin xmax*]将数据矩阵 *X* 转换为图像 *I*，*xmin* 对应灰度 0（最暗即黑），*xmax* 对应灰度 1（最亮即白）。如果不指定区间[*xmin xmax*]时，MATLAB 7.0 则自动将 *X* 矩阵中最小设为 *xmin*，最大设为 *xmax*。

值得注意的是输入矩阵 *X* 和输出图像 *I* 都是双精度类型。实际上，mat2gray 函数阈 imshow 函数功能类似，这一点将在后面进行论述。

11.2 图像的显示

图像的显示过程是将数字图像从一组离散数据还原为一幅可见图像的过程。严格地说，图像的显示在图像处理（尤其是图像分析过程）中并不是必须的，因为图像处理和分析过程都是基于图像数据的计算，以数字数据或决策的形式给出处理或分析的结果，其中间过程并不一定要求可视。但是图像的显示是提高图像处理分析性能非常有用的手段，通过图像的显示，可以监视图像处理过程，并与处理分析交互地控制处理分析过程。

图像显示最重要的特性是图像的大小、光度分辨率、灰度线性、平坦能力和噪声特性等，这些显示特性将共同决定一个数字图像显示系统的质量及其在特定应用中的适用性等性能指标。本节主要介绍 MATLAB 7.0 软件图像显示工具，MATLAB 7.0 及图像处理工具箱的显示功能非常强大，不仅可以用来显示各种类型的图像，还可以用多种方式显示图像及图像序列。本节将介绍 MATLAB 7.0 中的基本图像显示技术，包括多图像显示和纹理映射等。

11.2.1 标准图像显示技术

MATLAB 7.0 显示图像的主要方法是调用 image 函数，该函数可创建一个句柄图形图像对象，并且包含设置该对象的各种属性的调用语法；此外，还提供了与 image 函数类似的 imagesc 函数，利用该函数可以实现对输入图像数据的自动缩放。同时，还包含了一个附件的显示函数，即 imshow 函数，与 image 和 imagesc 函数类似，imshow 函数可用于创建句柄图形图像对象。此外，该函数也可以自动设置各种句柄属性和图像特征，以优化效果。

1．imshow 函数

当用户调用 imshow 函数显示图像时，将自动设置图形窗口、坐标轴和图像属性，以控制图像数据在 MATLAB 7.0 的解释方式。这些自动设置的属性包括图像对象的 CData 属性和 CDdata-Mapping 属性、坐标轴对象的 Clim 属性和图像窗口对象的 Colormap 属性。

在 MATLAB 7.0 中，imshow 函数的语法如下：

- imshow(I,n)；
- imshow(I,[low,high])；
- imshow(BW)；
- imshow(…,display_option)；
- imshow(x,y,A,…)；
- imshow filename；
- h = imshow(…)。

根据用户使用参数的不同和 MATLAB 7.0 工具箱的设置，imshow 函数在调用时除了完成前面提到的属性设置外，还可以设置其他的图形窗口对象和坐标轴对象的属性以定制显示效果、设置图像边框的隐藏属性等。

2．显示索引图像

利用 imshow 函数显示 MATLAB 7.0 的索引图像时，可以同时指定图像的数据矩阵和颜色映射表，具体调用形式为：

- imshow（X,map）：其中，对于 *X* 中的每一个像素，imshow 都将其显示为存储在 *map* 映射表矩阵的相应的行所对应的颜色。

imshow 函数将同时设置下面的一些用以控制显示颜色的句柄图形的属性，例如将图像的 CData 属性值设置为 *X* 矩阵中的数据，将图像的 CDataMapping 属性值设置为 direct，使坐标轴对象的属性失效，将图形窗口对象的 Colormap 属性设置为 map 矩阵中的数据。

3．显示灰度图像

调用 imshow 函数显示灰度图像的语法如下：

- imshow（I）：其中 *I* 为灰度图像的数据矩阵；
- imshow（I,N）：其中 *I* 为灰度图像的数据矩阵，*N* 为整数，用于指定对应灰度颜色映射表中的索引数。

4．显示二进制位图

下面介绍 imshow 函数显示二进制位图的语法。

- imshow（BW）：其中 *BW* 为二进制位图的数据矩阵。

如果该位图的图像矩阵属于类 double，则 imshow 函数将其视为灰度图来对待，同时将 CDataMapping 的属性值设置为 scaled；Clim 的属性值设置为[0 1]；Colormap 颜色映射表属性设置为灰度颜色映射表。在这种情况下，图像数据矩阵中值 0 所对应的像素显示为黑色，值 1 对应的像素点显示为白色。

5．显示 RGB 图像

RGB 图像即真彩图像。RGB 图像直接表征像素颜色，而不是其他像图像那样通过颜色映射表来指定像素颜色，显示 RGB 图像的语法如下：

- imshow(RGB)：其中 RGB 为一个 $m \times n \times 3$ 的图像数据阵列。在 MATLAB 7.0 中，该数据阵列属于类 double、类 uint8 或 uint16。数据阵列中元素的取值取决于该矩阵所属的类型，如果该数据阵列属于类 double，则其元素的取值范围是[0,1]；如果该数据矩阵属于类 uint8，则其元素的取值范围是[0,255]；如果该数据阵列属于类 uint16，则其元素的取值范围是[0,65535]。

6．显示图形文件中的图像

通常情况下，在显示图像时，该图像的对象数据保存在 MATLAB 7.0 运行内存中的一个或多个变量中。但是，如果用户将图像保存在可以通过 imread 函数读取的图形文件中，则可以通过下面的语法直接将其显示出来。

- imshow filename：*filename* 是需要打开的图形文件的路径以及文件名。

如果图像是多帧的，那么 imshow 将仅仅显示第一帧，这种调用格式对于图像扫描非常有用。

> 注意：在使用这种格式时，该图形文件必须在当前目录下，或在 MATLAB 7.0 目录下。如果图像数据没有保存在 MATLAB 7.0 工作平台中，可以通过使用 getimage 函数将从当前的句柄图形图像对象中获取图像数据。

11.2.2 特殊图像显示技术

在 MATLAB 7.0 的图像处理工具箱中，除了 imshow 函数外，还提供了一些实现特殊显示功能的函数。它们与 MATLAB 7.0 自身提供的图形函数相结合，为图像显示提供了各种特殊的显示技术，包括图像显示中添加颜色条；显示多帧图像阵列；将图像纹理映射到表面对象上。下面将具体介绍这些技术的实现方法。

1．添加颜色条

在 MATLAB 7.0 的图像显示中，可以利用 colorbar 函数将颜色条添加到坐标轴对象中。如果该坐标轴对象包含一个图像对象，则添加的颜色条将指示出该图像中不同颜色的数据值。

例如，下面的代码将首先过滤一个类为 uint8 的图像，然后将其显示为灰度图，并添加颜色条，示例代码如下：

```
RGB = imread('saturn.png');
I = rgb2gray(RGB);
h = [1 2 1; 0 0 0; -1 -2 -1];
I2 = filter2(h,I);
imshow(I2,[]), colorbar
```

执行结果如图 11-5 所示，从图中可以看出数值与颜色的对应关系。

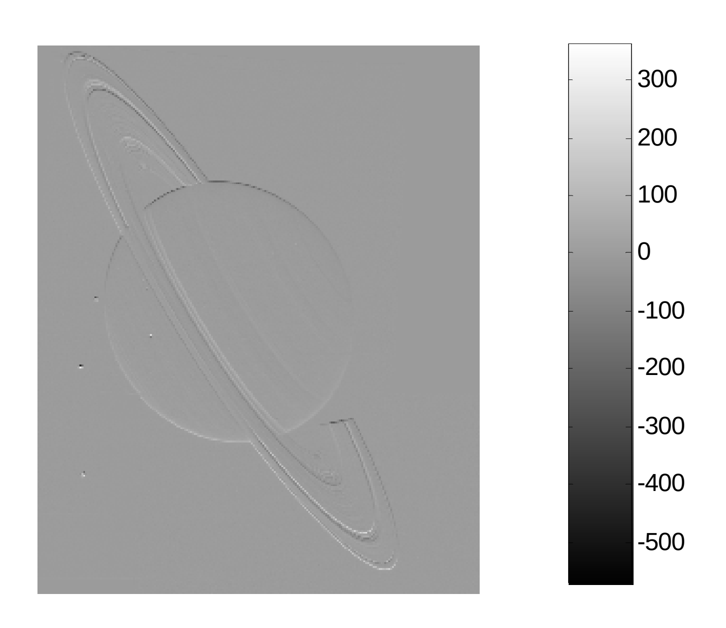


图 11-5　添加颜色条的灰度图

2．显示多帧图像阵列

多帧图像是一个包含多个图像的图像文件。MATLAB 7.0 支持的多帧图像的文件格式包括 HDF 和 TIFF 两种。文件一旦被读入 MATLAB，多帧图像的显示帧数由矩阵的第四维数值来决定。调用 montage 函数可实现多帧显示，该函数的语法如下：

- montage(I)；
- montage(BW)；
- montage(X,map)；
- h = montage(…)。

上面几种函数调用方式分别对应着不同的图像格式，有兴趣的读者可以一一验证。这里举一个简单的例子进行说明：

```
%定义一个 4 维矩阵，用来存储 27 幅核磁共振图像
mri = uint8(zeros(128,128,1,27));
%循环读出多帧图像中的每一幅图像
for frame=1:27
    [mri(:,:,:,frame),map] = imread('mri.tif',frame);
end
%多帧显示
montage(mri,map);
```

其运行结果如图 11-6 所示。

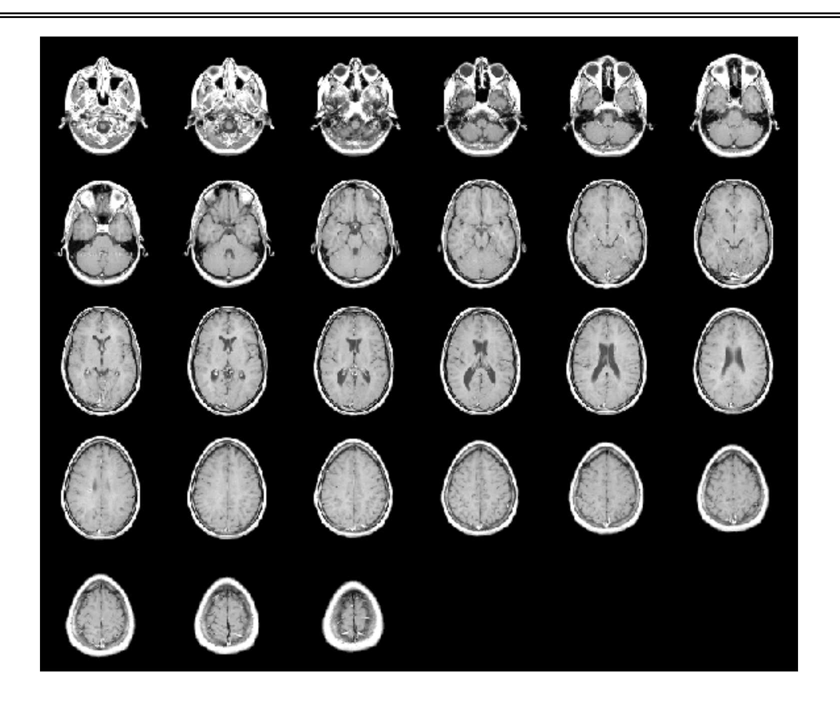


图 11-6　多帧显示实例

除了多帧显示之外，还可以利用 immovie 函数，从多帧图像阵列中创建 MATLAB 7.0 电影动画。值得提醒的是该函数只能用于索引图像，所以如果希望将其他类型的图像阵列转化为电影动画，则首先必须将该图像类型转换为索引类型。

3．纹理映射

在使用 imshow 函数时，MATLAB 7.0 在二维空间中显示图像。除此之外，MATLAB 7.0 专门提供了一个对图像进行纹理映射处理的函数 warp，使之显示在三维空间中，三维的面可以是柱面、球面以及自定义的三维曲面。warp 函数的语法格式如下：

- warp(X,map)；
- warp(I,n)；
- warp(BW)；
- warp(RGB)；
- warp(z,…)；
- warp(x,y,z,…)。

在 MATLAB 7.0 中，纹理映射是利用双线性渐进算法将图像映射到某个表面栅格上。例如下面的代码：

```
[x,y,z] = cylinder;
I = imread('testpat1.png');
warp(x,y,z,I);
```

将 testpat1.png 映射到圆柱体表面上，如图 11-7 所示。

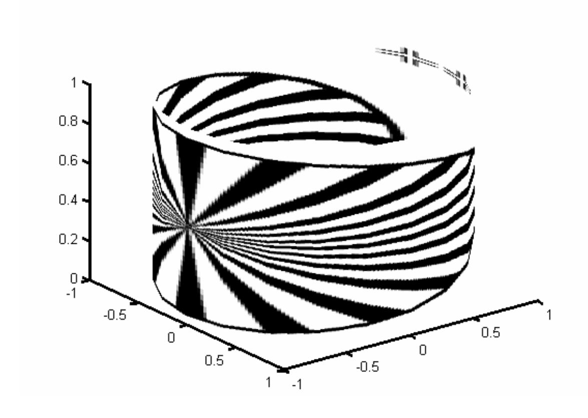


图 11-7 图像的纹理映射

有时，图像可能不是按照所期望的形式进行纹理映射的，此时可以对纹理映射的外观进行修改，其方法之一就是修改坐标轴的 *Xdir*、*Ydir* 和 *Zdir* 属性值。具体的修改方法，有兴趣的读者可以查看相关资料。

11.3 图像的几何运算

在处理图像的过程中，有时需要对图像的大小和几何关系进行调整，比如对图像进行缩放及旋转，这时图像中每个像素的值都要发生变化。数字图像的坐标是整数，经过这些变换之后的坐标不一定是整数，因此要对变换之后的整数坐标位置的像素进行估计。MATLAB 7.0 提供了一些函数实现这些功能。本节主要介绍 MATLAB 7.0 图像处理工具箱中的几何运算函数，这些函数支持所有的图像类型，可实现对图像进行缩放、旋转和剪裁等几何操作。

11.3.1 图像插值

插值是常用的数学运算，通常是利用曲线拟合的方法，通过离散的采样点建立一个连续函数来逼近真实曲线，用这个重建的函数便可以求出任意的函数值。设已知函数值为 w_1，w_2，…，则未知点 x 的函数值通过插值可以表示为：

$$f(x)=\sum_{l=1}^{L} w_l h(x-x_l)$$

其中 h（）为插值核函数，w_l 为权系数。插值算法的数值精度及计算量与插值核函数有关，插值核函数的设计是插值算法的核心。MATLAB 7.0 中的 imresize 函数核 imrotate 函数用于二维图像的插值。MATLAB 7.0 图像处理工具箱提供了 3 种插值方法：

1．最近邻插值（Nearest-neighbor interpolation）

这是一种最简单的插值方法，在这种算法中，每一个插值输出像素的值就是在输入图像中与其最临近的采样点的值。该算法的表达式如下：

$$f(x)=f(x_k) \qquad \frac{1}{2}(x_{k-1}+x_k)<x<\frac{1}{2}(x_k+x_{k+1})$$

最近临近插值是工具箱函数默认使用的插值方法，而且这种插值方法的运算量非常小。对于索引图像来说，它是惟一可行的方法。不过，最近临近插值法中的核函数的频域特性不好，从它的傅立叶谱上可以看出，它与理想低通滤波器的性质相差较大。当图像含有精细内容，也就是高频分量时，在用这种方法进行处理过的图像中可以明显看出块状效应。

2．双线性插值（Bilinear interpolation）

该方法输出像素值是它在输入图像中 2×2 领域采样点的平均值，它根据某像素周围 4 个像素的灰度值在水平和垂直两个方向上对其插值。

设 $m<i^{'}<m+1$，$n<j^{'}<n+1$，$a=i^{'}-m$，$b=j^{'}-n$，$i^{'}$，$j^{'}$ 是要插值点的坐标，则双线性插值的公式如下：

$$\begin{aligned}g(i^{'},j^{'})=(1-a)(1-b)g(m,n)+a(1-b)g(m+1,n)+\\(1-a)bg(m,n+1)+abg(m+1,n+1)\end{aligned}$$

把按上式计算出来的值赋予图像对策几何变换对应于 $(i^{'},j^{'})$ 处的像素，即可实现双线性插值。

3．双立方插值（Bicubic interpolation）

这种插值核为三次函数，其插值领域的大小为 4×4。它的插值效果比较好，但相应的计算量大。

这 3 种插值方法的运算方式基本类似。对于每一种方法来说，为了确定插值像素点的数值，必须在输入图像中查找与输出像素相对应的点。这 3 种插值方法的区别在于对像素点赋值的不同。其中，近邻插值输出像素的赋值为当前点的像素点；双线性插值输出像素的赋值为 2×2 矩阵所包含的有效点的加权平均值；双立方插值输出像素的赋值为 4×4 矩阵所包含的有效点的加权平均值。

11.3.2 图像大小调整

利用 imresize 函数通过一种特定的插值方法可以实现图像大小的调整。该函数的语法如下：

- B = imresize(A,m,method)；
- B = imresize(A,[mrows ncols],method)；
- B = imresize(…,method,n)；
- B = imresize(...,method,h)。

这里参数 method 用于指定插值的方法，可选的值为*'nearest'*、*'bilinear'*和*'bicubic'*，如果没有指定插值方法，则该函数将采用缺省的近邻插值（*nearest*）方法。

其中第 1 种语法返回图像大小等于 *A* 的大小乘以放大系数 *m*，若放大系数 *m* 设置在 0 到

1 之间，则 *B* 比 *A* 小，即图像缩小；如果放大系数 *m* 设置在大于 1，则图像放大。

第 2 种语法返回一个 *mrows* 行、*ncols* 列的图像，若 *mrows* 核 *ncols* 定义的长度比原图不同，则图像会产生变形。

在使用 bilinear 和 bicubic 方法缩小图像时，为消除引入的高频成分，imresize 使用一个前端滤波器，默认的滤波器尺寸为 11×11。也可以通过参数 *n* 指定滤波器的尺寸，即为上述 3 种语法结构。对于 nearest 插值方法，imresize 不使用前端滤波器，除非函数明确指定。

第 4 种语法结构是使用用户设计的插值核 *h* 进行插值，*h* 可以看作一个二维 FIR 滤波器。

下面是使用不同的插值方法对图像进行放大的程序：

```
load woman2
subplot(2,2,1)
imshow(X,map)
X1 = imresize(X,2,'nearest');
subplot(2,2,2)
imshow(X1,[]);
X2 = imresize(X,2,'bilinear');
subplot(2,2,3)
imshow(X2,[]);
X3 = imresize(X,2,'bicubic');
subplot(2,2,4)
imshow(X3,[]);
```

运行结果如图 11-8 所示。由图可见，在进行小倍数放大时，最近相邻插值方法的效果还可以，双线性插值方法的结果有些模糊，双立方插值效果最好。



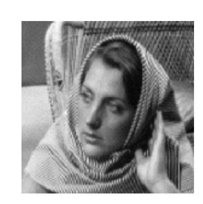



图 11-8　3 种不同的插值方法对图像进行放大示例

11.3.3　图像旋转

在对数字图像进行旋转的时候，各像素的坐标将会发生变化，使得旋转之后不能正好落在整数坐标外，需要进行插值。在工具箱中的函数 imrotate 可以用上述 3 种方法对图像进行

插值旋转，利用 imrotate 函数可以通过一种特定的插值方法来改变图像的显示角度。下面介绍该函数的语法格式。

- B = imrotate(A,angle,method)

使用指定的插值方法（通 imresize 函数的插值方法）逆时针方向将图像 *A* 旋转 *angle* 角度。返回图像 *B* 通常大于 *A*，包含整个旋转图像。若对图像进行顺时针旋转，则 *angle* 取负值。

一般来说，旋转后的图像会比原来图像大，超出原来图像的部分值为 0，为了使返回的图像与原来图像大小相同，可采用如下的格式：

- B = imrotate(A,angle,method,'crop')

其功能是通过指定 *crop* 参数对旋转后的图像进行剪切（取图像的中间部分），把图像进行 *angle* 角度旋转，然后返回和 *A* 大小相同的中间部分。

下面是将 circuit.tif 图像插值旋转 35° 的程序清单：

```
I = imread('circuit.tif');
J = imrotate(I,35,'bilinear');
subplot(1,2,1)
imshow(I)
subplot(1,2,2)
imshow(J)
```

其运行结果如图 11-9 所示。

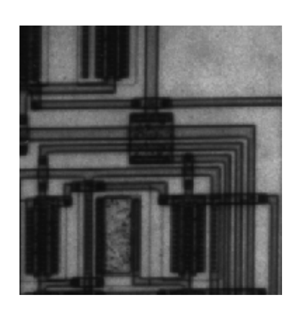
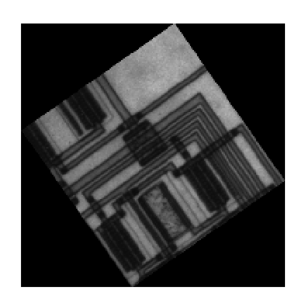



图 11-9　对图像进行插值旋转实例

11.3.4　图像剪裁

在图像处理过程中，有时只需要处理图像中的一部分，或者需要将某部分取出，这样就需要对图像进行剪切。图像处理工具箱中的 imcrop 函数将图像剪裁成指定矩形区域。该函数的语法如下：

- I2 = imcrop(I)；
- X2 = imcrop(X,map)；
- RGB2 = imcrop(RGB)。

上面这 3 种语法的功能是交互地对灰度图像、索引图像和真彩图像进行剪切，显示图像，

允许用鼠标指定剪裁矩形。

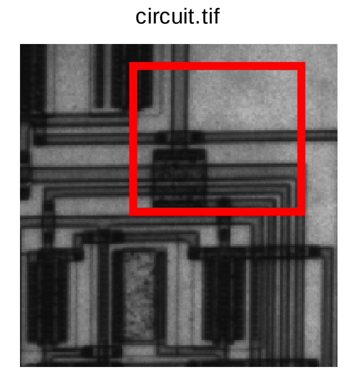
- I2 = imcrop(I,rect)
- X2 = imcrop(X,map,rect)
- RGB2 = imcrop(RGB,rect)

上面这三种语法的功能是非交互式指定剪裁矩阵，按指定的矩阵框 *rect* 剪切图像，*rect* 为四元素向量[xmin ymin width height]，分别表示矩形的左下角、右下角、长度和宽度，这些值在空间坐标中指定。

下面是从 ic.tif 图像中减去鼠标左键拖动选取的举行区域，并以新的图形窗口显示的程序：

```
subplot(1,2,1)
imshow circuit.tif
I = imcrop;
subplot(1,2,2)
imshow(I);
```

其运行结果如图 11-10 所示。



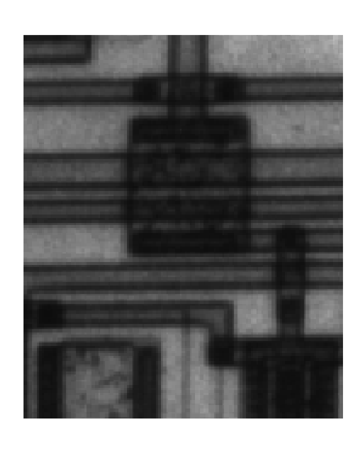


图 11-10　图像进行剪切示例

11.4　图像的变换技术

为了快速有效地对图像进行处理和分析，常常需要将原定义在图像空间的图像以某种形式转换到另外一些空间，并利用这些空间特有的性质方便地进行一定的加工，最后再转换回图像空间以得到需要的效果。这种使图像处理简化的方法通常是对图像进行变换。图像变换技术在图像增强、图像恢复和有效地减少图像数据、进行数据压缩以及特征提取等方面都有着十分重要的作用。本节将主要对应用最多傅立叶变换、离散余弦变换、小波变化以及 MATLAB 7.0 实现进行比较详细的介绍。

11.4.1 数字图像的二维傅立叶变换

在图像处理的广泛领域中，傅立叶变换起着非常重要的作用，包括图像的效果增强、图像分析、图像复原和图像压缩等。在图像数据的数字处理中常用的是二维离散傅立叶变化，它能把空间域的图像转变到频域上进行研究，从而能很容易地对图像的各空间频域成分进行相应处理。在第 10 章已经比较详细地介绍了傅立叶变换以及离散傅立叶变换的概念，而二维傅立叶变换又是一维傅立叶变换的简单的推广，这里不再赘述，着重介绍二维傅立叶变换在图像处理中的应用。

1．MATLAB 7.0 提供的快速傅立叶变换函数

在 MATLAB 7.0 中，提供了 fft 函数、fft2 函数和 fftn 函数分别用于进行一维 DFT、二维 DFT 和 N 维 DFT 的快速傅立叶变换；ifft 函数、ifft2 函数和 ifftn 函数分别用于进行一维 DFT、二维 DFT 和 N 维 DFT 的快速傅立叶反变换。下面分别进行具体介绍。

（1）fft2 函数

该函数是用于计算二维快速傅立叶变化，其语法格式为：

- B = fft2(I)：返回图像 *I* 的二维 fft 变换矩阵，输入图像 *I* 和输出图像 *B* 大小相同；
- B = fft2(I,m,n)：通过对图像 *I* 剪切或补零，按用户指定的点数计算 *fft*，返回矩阵 *B* 的大小为 $m \times n$。很多 MATLAB 7.0 图像显示函数无法显示复数图像，为了观察图像傅立叶变换后的结果，应对变换后的结果求模，方法是对变换结果调用 *abs* 函数。

（2）fftn 函数

该函数用于 *n* 维傅立叶变换，其语法格式为：

- B = fftn(I)：计算图像的 *n* 维傅立叶变换，输出图像 *B* 与输入图像 *I* 大小相同；
- B = fftn(I,size)：函数通过对图像 *I* 剪切或补零，按 *size* 指定的点数计算给定矩阵 *n* 维傅立叶变换，返回矩阵 *B* 的大小也是 *size*。

（3）fftshift 函数

该函数是用于将变换后图像频谱中心从矩阵的原点移到矩阵的中心，其语法格式为：

- B = fftshift(I)：可以用于调整 *fft*、*fft2* 和 *fftn* 的输出结果。对于向量，fftshift(I)将 *I* 的左右两半交换位置；对于矩阵 *I*，fftshift(I)将 *I* 的一、三象限和二、四象限进行互换；对于高维矢量，fftshift(I)将矩阵各维的两半进行互换。

（4）ifft2 函数

该函数用于计算图像的二维傅立叶反变换，其语法格式为：

- B = ifft(I)：返回图像 *I* 的二维傅立叶反变换矩阵，输入图像 *I* 和输出图像 *B* 大小相同；
- B = ifft2(I,m,n)：通过对图像 I 剪切或补零，按用户指定的点数计算二维傅立叶反变换，返回矩阵 ***B*** 的大小维 $m \times n$、通常输出矩阵 ***B*** 为复数矩阵，如果要求模，需要调用 abs 函数。

（5）ifftn 函数

该函数用于计算 *n* 维傅立叶反变换，其语法格式为：

- B = ifftn(I)：计算图像的 *n* 维傅立叶反变换矩阵，输出图像 *B* 与输出图像 *I* 大小相同；

- B = ifftn(I,size)：函数通过对图像 *I* 剪切或补零，按 *size* 指定的点数计算给定矩阵 *n* 维傅立叶反变换，返回矩阵 *B* 的大小也是 *size*。

2．二维傅立叶变换的 MATLAB 7.0 实现

下面，举例来说明傅立叶变换的实现语句 B = fft2(A)，该语句执行对矩阵 A 有二维傅立叶变换。给出一幅图像（saturn2.tif），其傅立叶变换程序如下：

```
figure(1);
load imdemos saturn2;
%显示图像
imshow(saturn2);
figure(2);
%进行傅立叶变换
B = fftshift(fft2(saturn2));
%显示变换后的系数分布
imshow(log(abs(B)),[]),colormap(jet(64)),colorbar;
```

运行结果如图 11-11 所示。



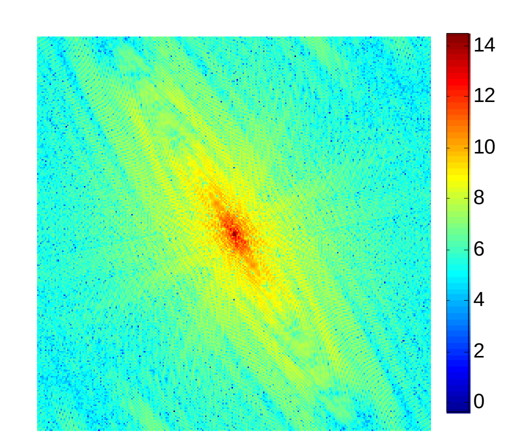


图 11-11　二维傅立叶变换图

3．滤波器频率响应

利用傅立叶变换可以得到线性滤波器的频率响应，其过程如下：首先求出滤波器的脉冲响应，然后利用快速傅立叶变换算法对滤波器的脉冲响应进行变换，得到的结果就是线性滤波器的频率响应。MATLAB 7.0 工具箱中提供的 freqz2 函数就是利用这个原理可以同时计算和显示滤波器的频率响应。

下面是一个利用 freq2 函数得到的高斯滤波器的频率响应的程序：

```
h = fspecial('gaussian');
freqz2(h)
```

运行结果如图 11-12 所示。

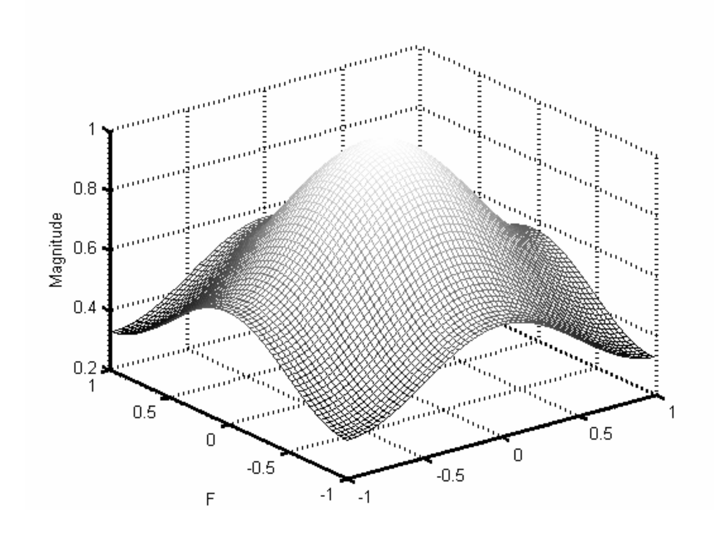


图 11-12　高斯低通滤波器

4．快速卷积

傅立叶变换的另一个重要特性就是能够实现快速卷积。由线性系统理论可知，两个函数卷积的傅立叶变换等于两个函数的傅立叶变换的乘积。该特性与快速傅立叶变换相结合，可以快速计算函数的卷积。假设 A 是一个 $M \times N$ 的矩阵，B 是一个 P×Q 的矩阵，则快速计算矩阵的方法如下：

- 对 A 和 B 进行零填充，将 A 和 B 填充为 2 的幂次矩阵；
- 使用 fft 计算 A 和 B 的二维 DFT；
- 将两个 DFT 计算结果相乘；
- 使用 ifft2 计算上一步所得的二维 DFT 的反变换。

下面是一个计算魔方阵和一个 1 矩阵的卷积的程序：

```
A = magic(3);
B = ones(3);
%对 A 进行零填充，使之成为 8×8 矩阵
A(8,8) = 0;
%对 B 进行零填充，使之成为 8×8 矩阵
B(8,8) = 0;
C = ifft2(fft2(A).*fft2(B));
%抽取矩阵中的非零部分
C = C(1:5,1:5);
%去掉错误的，由四舍五入产生的虚部
C = real(C)
```

运行结果如下：

```
C =
```

```
     8.0000    9.0000   15.0000    7.0000    6.0000
    11.0000   17.0000   30.0000   19.0000   13.0000
    15.0000   30.0000   45.0000   30.0000   15.0000
     7.0000   21.0000   30.0000   23.0000    9.0000
     4.0000   13.0000   15.0000   11.0000    2.0000
```

5．图像特征识别

傅立叶变换还能够用来分析两幅图像的相关性，相关性可以用来确定一幅图像的特征，在这个意义下，相关性通常被成为模板匹配。例如，假如希望在图像 text.tif 中定位字符“*a*”，如图 11-13 左上所示，可以采用下面的方法定位。

将包含字母“a”的图像与 text.tif 图像进行相关运算，也就是首先将字母 *a* 和图像 text.tif 进行傅立叶变换，然后利用快速卷积的方法，计算字母 *a* 和图像 text.tif 的卷积（其结果如图 11-13 右上所示），提取卷积运算的峰值，如图 11-13 左下所示的白色亮点，即得到在图像 text.tif 中对字母“a”定位的结果。

程序代码如下：

```
%读入图像'text.tif'
bw = imread('text.png');
%从图像中抽取字母 a 的图像
a = bw(32:45,88:98);
subplot(2,2,1),
imshow(bw);
subplot(2,2,2),
imshow(a);
C = real(ifft2(fft2(bw) .* fft2(rot90(a,2),256,256)));
subplot(2,2,3)
imshow(C,[])
%选择一个略小于 C 中最大值的值做为阈值
thresh = 60; %
subplot(2,2,4),
%显示像素值超过阈值的点
imshow(C > thresh)%
```

程序运行结果如图 11-13 所示。

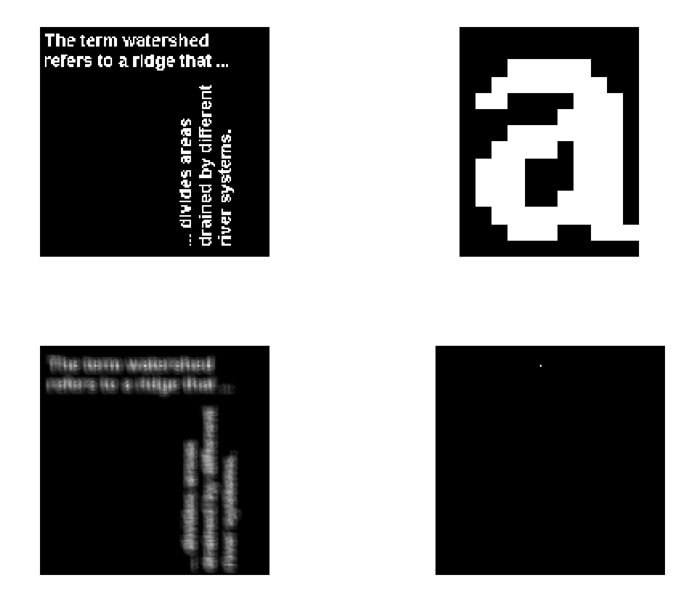


图 11-13 **图像特征识别示例**

11.4.2 数字图像的离散余弦变换

离散余弦变换的变换核为余弦函数，计算速度比较快，有利于图像雅座和其他处理。在大多数情况下，离散余弦变换（DCT）用于图像的压缩操作中。JPEG 图像格式的压缩算法采用的是 DCT。

1．**离散余弦变换的定义**

一维离散余弦正反变换公式如下：

$$F(k)=2\sum_{n=0}^{N-1}f(n)\cos\frac{\pi(2n+1)k}{2N}\qquad n,k=0,1,\cdots,N-1$$

$$f(n)=\frac{1}{N}\sum_{k=0}^{N-1}F(k)\cos\frac{\pi(2n+1)k}{2N}\qquad n,k=0,1,\cdots,N-1$$

类似于一位离散余弦变换，二维离散余弦正变换公式为：

$$F(u,v)=c(u)c(v)\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\cos\frac{\pi(2x+1)u}{2M}\cos\frac{\pi(2y+1)v}{2N}$$

$$u=0,1,\cdots,M-1;v=0,1,\cdots,N-1$$

其中：

$$c(u)=\begin{cases}\sqrt{1/M} & u=0\\ \sqrt{2/M} & u=1,2\Lambda\ ,M-1\end{cases}$$

$$c(v)=\begin{cases}\sqrt{1/N} & u=0\\ \sqrt{2/N} & u=1,2\Lambda\ ,N-1\end{cases}$$

二维离散余弦反变换公式为：

$$f(x,y)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}c(u)c(v)F(u,v)\cos\frac{\pi(2x+1)u}{2M}\cos\frac{\pi(2y+1)v}{2N}$$

$$x=0,1,\Lambda\ ,M-1;y=0,1,\Lambda\ ,N-1$$

其中 x，y 为空间域采样值，u，v 为频域采样值。通常数字图像用像素方阵标识，即 M = N。在这种情况下，二维离散余弦正反变换可以简化为：

$$F(u,v)=c(u)c(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\frac{\pi(2x+1)u}{2N}\cos\frac{\pi(2y+1)v}{2N}$$

$$u=0,1,\Lambda\ ,N-1;v=0,1,\Lambda\ ,N-1$$

$$f(x,y)=\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}c(u)c(v)F(u,v)\cos\frac{\pi(2x+1)u}{2N}\cos\frac{\pi(2y+1)v}{2N}$$

$$x=0,1,\Lambda\ ,N-1;y=0,1,\Lambda\ ,N-1$$

其中，$c(u)=\begin{cases}\sqrt{1/2} & u=0\text{或}v=0\\ 1 & u,v=1,2\Lambda\ ,M-1\end{cases}$

在 MATLAB 7.0 中，$c(u)c(v)F(u,v)\cos\frac{\pi(2x+1)u}{2N}\cos\frac{\pi(2y+1)v}{2N}$称为离散余弦变换的基础函数。这样 DCT 系数 $F(u,v)$ 可以看成是每一个基础函数的加权。例如，对应一个 8×8 矩阵，它的 64 个基础函数如图 11-14 所示。

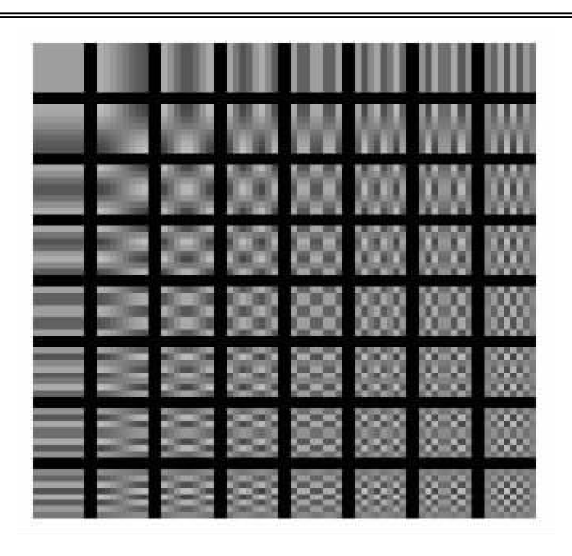


图 11-14　8×8 矩阵的 64 个基础函数

在图 11-14 中，这些基础函数的水平频率从左到右增长，垂直频率从上到下增长。位于图像左上方的定值基础函数通常被成为 DC 基础函数，相应的 DCT 系数称为 DC 系数。

2．MATLAB 7.0 提供的 DCT 变换函数

（1）dct2 函数

该函数用于实现图像的二维离散余弦变换，其语法格式为：

- B = dct2(A)：返回图像 A 的二维离散余弦变换值，它的大小与 A 相同，且各元素为离散余弦变换的系数 F(k1,k2)。
- B = dct2(A,m,n)
- B = dct2(A,[m,n])：在对图像 A 进行二维离散余弦变换之前，先将图像 A 补零至 $m×n$。如果 m 和 n 比图像 A 小，则进行变换之前，将图像 A 剪切。

（2）idec2 函数

该函数可以实现图像的二维离散余弦反变换，其语法格式为：

- B = idct2(A)：返回图像 A 的二维离散余弦反变换值，它的大小与 A 相同。
- B = idct2(A,m,n)
- B = idct2(A,[m,n])：在对图像 A 进行二维离散余弦反变换之前，先将图像 A 补零至 $m×n$。如果 n 和 n 比图像 A 小，则进行变换之前，将图像 A 剪切。

（3）dctmtx 函数

该函数用于计算二维离散余弦变换矩阵，其语法格式为：

- D = dctmtx(n)：返回 $n×n$ 的 DCT 变换矩阵，如果矩阵 A 的大小为 $n×n$，$D×A$ 是 A 矩阵每一列的 DCT 变换值，$D'×A$ 是 A 每一列的反变换值。如果矩阵 A 为 $n×n$ 的方阵，则 A 的 DCT 变换可以用 $D×A×D'$来计算。特别是对于 A 很大的情况，比利用 dct2 计算二维离散 DCT 变换要快。

3．离散余弦变换的 MATLAB 7.0 实现

在 MATLAB 7.0 中，函数 dct2 核函数 idct2 分别用于进行二维 DCT 变换和二维 DCT 反变换。下面举例来说明二维余弦正反变换在 MATLAB 7.0 中的实现，程序代码如下：

```
%载入图像
RGB = imread('autumn.tif');
figure(1);
imshow(RGB);
%将真彩图转换为灰度图
I = rgb2gray(RGB);
figure(2);
imshow(I);
%进行余弦变换
J = dct2(I);
figure(3);
imshow(log(abs(J)),[]);
colormap(jet(64));
colorbar;
%将 DCT 变换值小于 11 的元素设为 0
J(abs(J)<11) = 0;
%进行余弦反变换
K = idct2(J)/255;
figure(4);
imshow(K)
```

程序运行输入结果如图 11-15 所示，其中左上为原始图像，右上为对应的灰度图像，左下为余弦变换系数，右下为余弦反变换恢复图像。

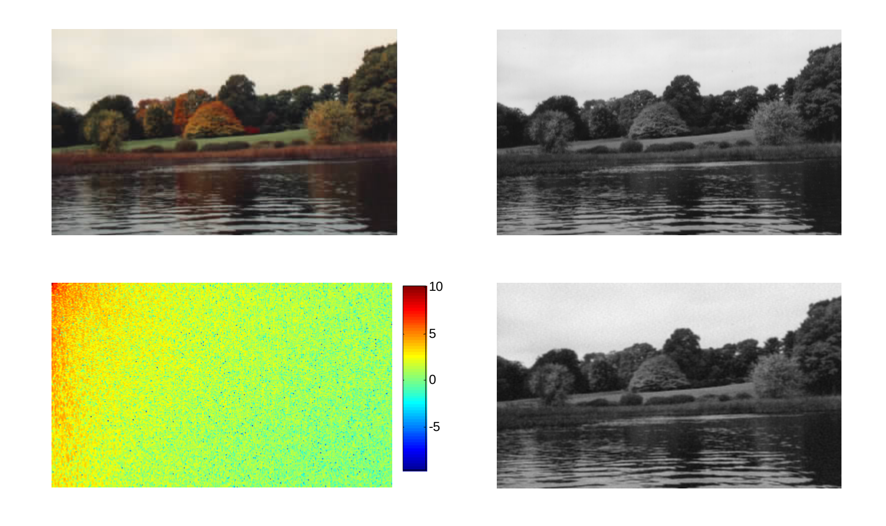


图 11-15　余弦变换与反变换图示

4．离散余弦变换的应用

离散余弦变换在图像压缩中具有广泛的应用，下面仅介绍一个用 DCT 压缩的例子。

在 JPEG 图像压缩算法中，首先将输入图像分解为 8×8 或 16×16 的图像块，然后对每个图像块进行二维 DCT 变换，最后将变换得到的量化的 DCT 系数进行变化或传送，形成压缩后的图像格式。在接收端，将量化的 DCT 系数进行解码，并对每一个 8×8 块或 16×16 块进行二维 DCT 反变换，最后将操作完成后的块组成一个单个的图像。至此，完成图像的压缩和解压缩过程。对于一幅典型的图像而言，大多数的 DCT 系数的值非常接近于 0，如果舍弃这些接近于 0 的 DCT 系数值，在重构图像时并不会因此而带来画面质量的显著下降。故利用 DCT 进行图像压缩可以节约大量的存储空间。

下面是一个展示了如何把如图 11-16 左边所示的输入图像划分成 8×8 的图像块，计算它们的 DCT 系数，并且只保留 64 个 DCT 系数中的 11 个，然后对每个图像块利用这 11 个系数进行 DCT 反变换来重构图像的程序：

```
I = imread('cameraman.tif');
I = im2double(I);
%产生二维 DCT 变换矩阵
T = dctmtx(8);
%计算二维 DCT
B = blkproc(I,[8 8],'P1*x*P2',T,T');
%二值掩模，用来压缩 DCT 系数
mask = [1    1    1    1    0    0    0    0
        1    1    1    0    0    0    0    0
        1    1    0    0    0    0    0    0
        1    0    0    0    0    0    0    0
        0    0    0    0    0    0    0    0
        0    0    0    0    0    0    0    0
        0    0    0    0    0    0    0    0
        0    0    0    0    0    0    0    0];
%只保留 DCT 变换的 11 个系数
B2 = blkproc(B,[8 8],'P1.*x',mask);
%DCT 反变换，用来重构图像
I2 = blkproc(B2,[8 8],'P1*x*P2',T',T);
subplot(1,2,1)
imshow(I),
subplot(1,2,2)
imshow(I2)
```

运行结果如图 11-16 所示。




图 11-16　离散余弦变换在图像压缩应用示例

11.4.3　其他变换技术

除了上面介绍的傅立叶变换和离散余弦变换外，在图像处理中还有几种使用比较广泛的变换技术，它们是离散沃尔什变换、离散哈达玛变换以及 Randon 变换。这里只简略的介绍每种变换的基本思想，详细的内容以及使用方法请有兴趣的读者参看书后的附录以及 MATLAB 7.0 自带的帮助文档。

1．离散沃尔什变换

傅立叶变换和离散余弦变换都是由正弦或余弦三角函数为基本的正交函数基，在快速算法中要用到复数乘法、三角函数乘法，占用时间仍然较多。在某些领域，需要有更为有效和便利的变换方法。沃尔什（Walsh）变换就是其中一种。它包括只有+1 和-1 两个数值所构成的完备正交基。由于沃尔什函数基就是二值正交基，与数字逻辑的两个状态相对应，因而更加适用于计算机处理。另外，与傅立叶变换相比，沃尔什变换减少了存储空间和提高了运算速度，这一点对图像处理来说是至关重要的。特别是在大量数据需要进行实时处理时，沃尔什变换更加显示出其优越性。

2．离散哈达玛变换

哈达玛（Hadamard）变换本质上是一种特殊排序的沃尔什变换，哈达玛变换矩阵也是一个方阵，只包括+1 和-1 两个矩阵元素，各行和各列之间彼此是正交的，即任意二行相乘或二列相乘后的各数之和必须为零。哈达玛变换核矩阵与沃尔什变换不同指出仅仅是行的次序不同。哈达玛变换的最大优点在于它的变换核矩阵具有简单的递推关系，即高阶矩阵可以用两个低阶矩阵求得。这个特点使人们更愿意采用哈达玛变换，不少文献中常采用沃尔什—哈达玛变换这一术语。

3．Radon 变换

Radon 变换是计算图像在某一角度射线方向上投影的变换方法。我们知道，二维函数 f(x,y)的投影是其在确定方向上的线积分。例如，f(x,y)在垂直方向上的二维线积分就是 f(x,y)在 x

轴上的投影；f(x,y)在水平上的二维线积分就是 f(x,y)在 y 轴上的投影。图 11-17 说明了一个简单二维函数的水平和垂直投影。

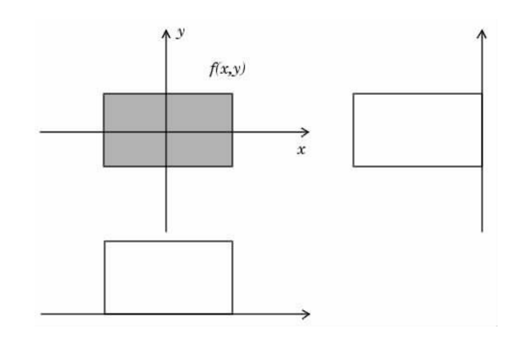


图 11-17　二维函数的水平投影核垂直投影示意图

另外，可以沿任意角度 θ 对函数进行投影，即任意角度函数 f(x,y)的 Radon 变换可以表达为：

$$R_\theta(x') = \int_{-\infty}^{\infty} f(x'\cos\theta - y'\sin\theta, x'\sin\theta + y'\cos\theta)dy'$$

其中 $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$，函数 f(x,y)的 Radon 变换几何示意图如图 11-18 所示。

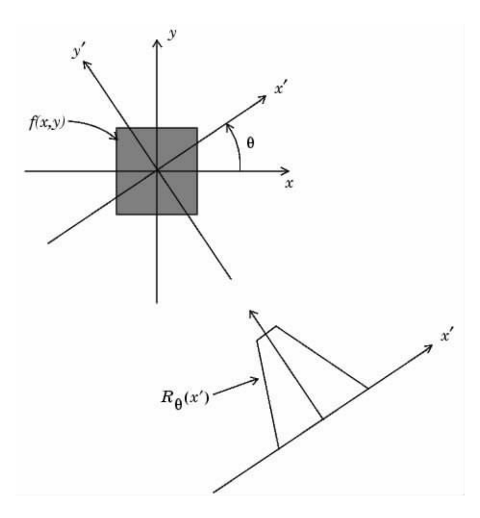


图 11-18　函数 f(x,y)的 Radon 变换几何示意图

11.5　图像分析

MATLAB 7.0 的图像处理工具箱支持多种标准的图像处理操作，以方便用户对图像进行分析和调整。这些图像处理操作主要包括：

- 获取像素值及其统计数据；

- 分析图像，抽取其主要结构信息；
- 调整图像，突出其某些特征或抑制噪声。

11.5.1　像素值及其统计

MATLAB 7.0 的图像处理工具提供多个函数以返回与构成图像的数据值相关的信息，这些函数能够以多种形式返回图像数据的信息，主要包括：

- 选定像素的数据值（pixval 函数和 impixel 函数）；
- 沿图像中某个路径的数据值（improfile 函数）；
- 图像数据的轮廓图（imcontour 函数）；
- 图像数据的柱状图（imhist 函数）；
- 图像数据的摘要统计值（mean2 函数、std2 函数和 corr2 函数）；
- 图像区域的特征度量（imfeature 函数）。

1．像素选择

图像处理工具箱中包含两个函数可以返回用户指定的图像像素的颜色数据值。

（1）pixval 函数

当光标在图像上移动时，该函数以交互的方式显示像素的数据值。另外，该函数还可以显示两个像素之间的 Euclidean 距离。

（2）impixel 函数

impixel 函数可以返回选中像素或像素集的数据值。用户可以直接将像素坐标作为该函数的输入参数，或用鼠标选中像素。

例如，在下面的例子中，首先调用 impixel 函数，然后在显示的 canoe.tif 图像中用鼠标选中 3 个点，代码如下：

```
imshow canoe.tif
vals = impixel
```

上面的代码运行后，得到如图 11-19 所示的运行界面。选中 3 个点后，按回车键，则命令行中得到的结果如下：

```
vals =
    0.1922    0.1922    0.1608
    0.5176         0         0
    0.4824    0.3882    0.2902
```

值得注意的是在所得的结果中，对应于第 2 个像素（该像素位于小船上）的值为纯红色，其绿色和蓝色成分均为 0；另外，对应索引图像，pixval 函数和 impixel 函数都将其显示为存储在颜色映像中的 RGB 值而不是索引值。

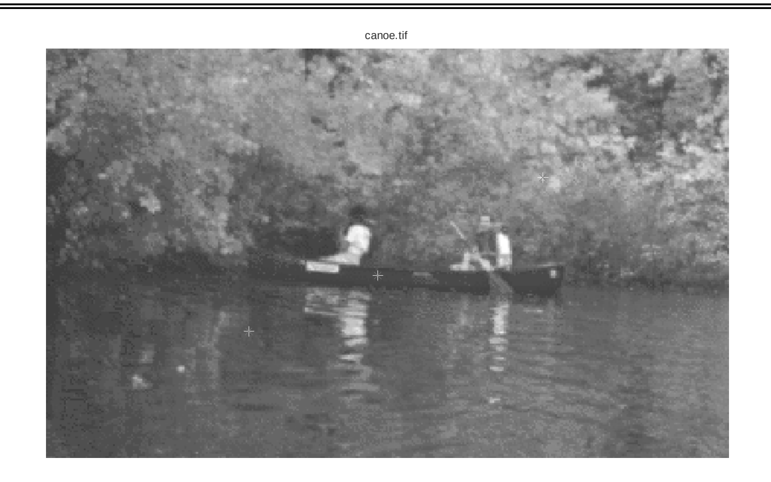


图 11-19　impixel 函数的运行界面

2．强度描述图

在 MATLAB 7.0 图像处理工具箱中提供了 improfile 函数用于沿着图像中一条直线段路径或直线路径计算并绘制其强度（灰度）值。例如下面的代码：

```
I = fitsread('solarspectra.fts');
imshow(I,[]);
improfile
```

运行后，得到如图 11-20 所示的运行界面。确定直线段或直线路径后，按回车键，则得到如图 11-21 所示的轨迹强度（灰度）图。

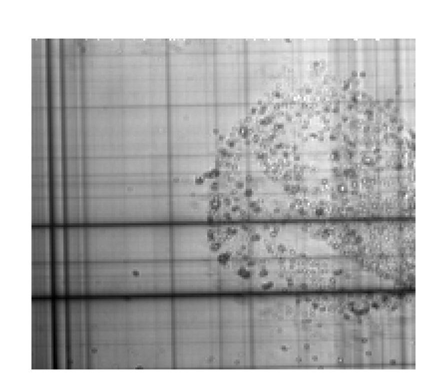


图 11-20　improfile 函数的运行界面（灰度图）

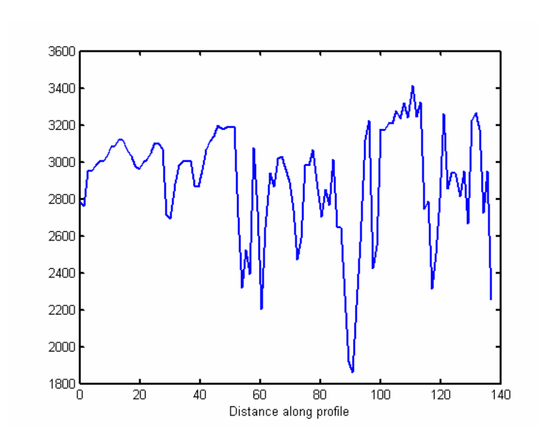


图 11-21　improfile 函数的运行结果

注意：图 11-21 中的峰值对应于图 11-20 中的黑色或白色。

下面的示例演示 improfile 函数如何处理 RGB 图像。具体的代码如下：

```
imshow peppers.png
improfile
```

运行后，得到如图 11-22 所示的运行界面。确定直线段或直线路径后，按回车键，则得

到如图 11-23 所示的轨迹强度图。



图 11-22　improfile 函数的运行界面（RGB）

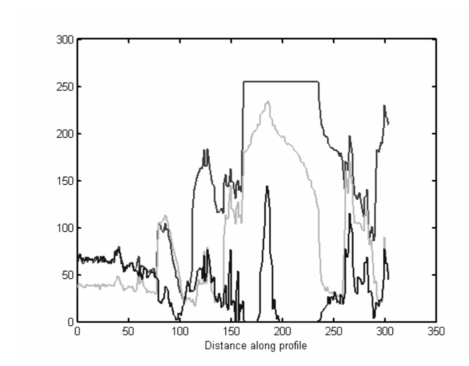


图 11-23　improfile 函数的运行结果（RGB）

由图 11-23 可以看出，improfile 函数绘制的强度图是将红色、绿色和蓝色分离开了，各自表达为独立的线条图形。

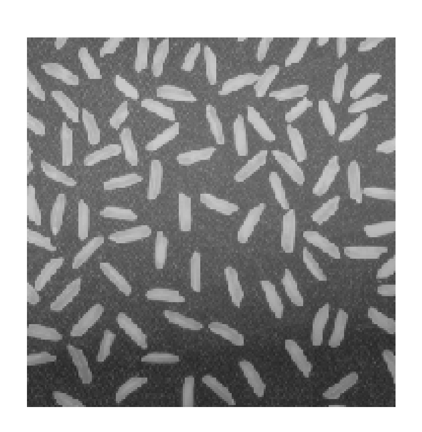
3．图像轮廓图

在灰度图的轮廓图显示时，可以利用 MATLAB 7.0 图像处理工具箱中的 imcontour 函数。该函数类似于 contour 函数，与 contour 函数相比，其功能更全。它能够自动设置坐标轴对象，从而使得其方向和纵横比能够与所显示的图形相匹配。

例如下面的代码：

```
I = imread('rice.png');
subplot(1,2,1)
imshow(I)
subplot(1,2,2)
imcontour(I,3)
```

运行后，分别在图形窗口中显示原始图像 rice.png 及其轮廓图，如图 11-24 所示。



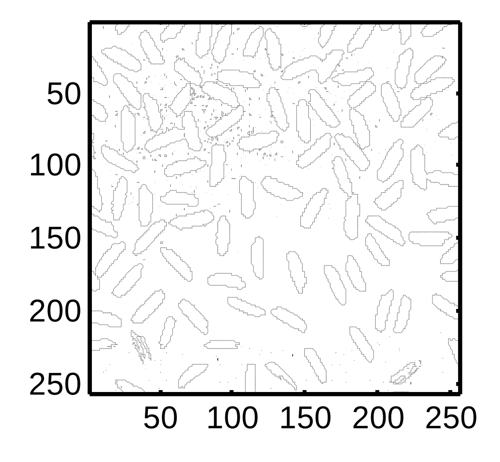


图 11-24　图像轮廓图

4．图像柱状图

图像柱状图可以用来显示索引图像或灰度图像中的灰度分布。可以利用 MATLAB 7.0 图像处理工具箱中的 imhist 函数创建柱状图。下面以前面介绍的大米灰度图为例来创建该图的柱状图。其代码设置如下：

```
I = imread('rice.png');
subplot(1,2,1)
imshow(I)
subplot(1,2,2)
imhist(I,64)
```

代码运行后的结果如图 11-25 所示。由此可见，柱状图的峰值出现在 100 附近，这是因为大米堆的背景色为深灰色所致。

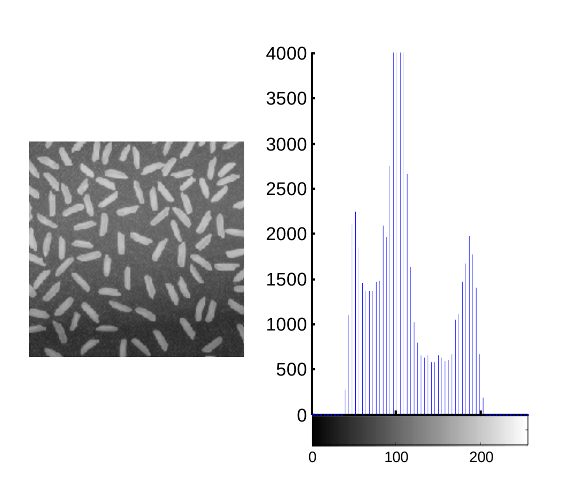


图 11-25　大米灰度图的柱状图

11.5.2　图像分析

在 MATLAB 7.0 中的图像分析技术可以提取图像的结构信息。例如，可以利用图像处理工具箱中提供的 edge 函数来探测边界。这里所谓的边界，其实就是图像中包含的对象所对应的位置。下面介绍几种常见图像分析函数。

1．灰度图像的边缘：edge 函数

该函数的语法如下：

- BW = edge(I,method)：返回与 *I* 大小一样的二进制图像 *BW*，其中元素 1 为发现 *I* 中的边缘。*Method* 为下列字符串之一。

'sobel'　　缺省值，用导数的 sobel 近似值检测边缘，那些梯度最大点返回边缘；
'prewitt'　用导数的 prewitt 近似值检测边缘，在那些梯度最大点返回边缘；
'roberts'　用导数的 roberts 近似值检测边缘，在那些梯度最大点返回边缘；

'log'　　使用高斯滤波器的拉普拉斯运算对 I 进行滤波，通过寻找 0 相交检测边缘；
'zerocross' 使用指定的滤波器对 I 滤波器后，寻找 0 相交检测边缘。

- BW = edge(I,method,thresh)：用 *thresh* 指定灵敏度阈值，所有不强于 *thresh* 的边缘都被忽略；
- BW = edge(I,method,thresh,direction)：对于*'sobel'*和*'prewitt'*方法指定方向。该函数中的 *direction* 为字符串，即*'horizontal'*标识水平方向，*'vertical'*表示垂直方向，*'both'*两个方向（缺省值）；
- BW = edge(I,'log',thresh,sigma)：用 *sigma* 指定标准差。

2．行四叉树分解：qtdecomp 函数

将一块图像分成 4 块等大小的方块，然后判断每块是否满足同性质的标准，如果满足，则不再分解，否则，再进行分成 4 块，并对每块应用测试标准，分解过程重复迭代下去，直到满足标准。结果可能包含不同大小的块，改函数的语法结构如下：

- S = qtdecomp(I)：对灰度图像 *I* 进行四叉树分解，返回四叉树结构的稀疏矩阵 *S*。
- S = qtdecomp(I,threshold)：如果块中元素最大值减去元素最小值大于 *threshold*，则分解块。*Threshold* 为 0～1 之间的值。
- S = qtdecomp(I,threhold,mindim)：如果块小于 *mindim* 就不再进行分解，无论其符合阈值条件与否。
- S = qtdecomp(I,FUN)：使用函数 *FUN* 确定是否分解块。

3．获取四叉树分解块值：qtdgetblk 函数

该函数的语法结构如下：

- [VALS,R,C] = qtdgetblk(I,S,dim)：*VALS* 中对应 dim×dim 块的取值取代 *I* 的四叉树分解中的每个 dim×dim 块。*S* 为 qtdecomp 函数返回的稀疏矩阵，包含四叉树结构；VALS 是 dim×dim×*k* 数组，*k* 是四叉树分解的 dim×dim 块的数量。如果没有指定大小的块，则返回一个空矩阵。*R* 和 *C* 为包含块左上角行列坐标的向量。
- [VALS,IDX] = qtdgetblk(I,S,dim)：返回块左上角直线索引的向量 *IDX*。

4．设置四叉树分解块值：qtsetblk 函数

- J = qtsetblk(I,S,dim,VALS)：用 *VALS* 中对应 dim×dim 块的值取代 *I* 的四叉树分解中的每个 dim×dim 块。*S* 为 qtdecomp 函数返回的稀疏矩阵，包括四叉树结构；*VALS* 是 dim×dim×*k* 数组，*k* 是四叉树分解的 dim×dim 块数量。

5．实例

（1）图像分析中的灰度边缘检测实例

源代码设置如下：

```
%调入与显示 RGB 图像
RGB = imread('peppers.png');
isrgb(RGB);
```

```
figure(1);
imshow(RGB);
%RGB 图转换为灰度图像
I = rgb2gray(RGB);
figure(2);
imshow(I);
colorbar('horiz');
isgray(I);
%边缘检测
ED = edge(I,'sobel',0.08);
figure(3)
imshow(ED);
```



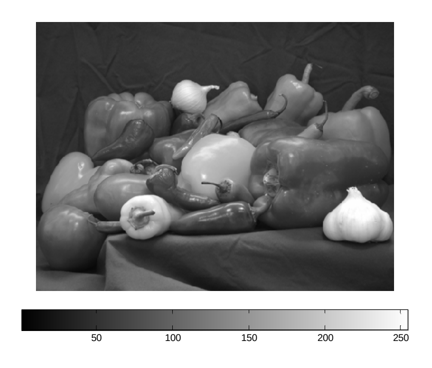




图 11-26　灰度边缘检测实例

运行结构如图 11-26 所示，图（a）为原始图像，图（b）为对应的灰度图，图（c）为检测到的灰度边缘。

（2）Sobel 边界探测器和 Canny 边界探测器再图像分析中的应用实例

源代码设置如下：

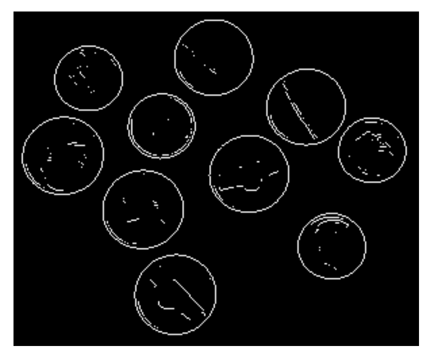
```
%读入原始图像
I = imread('coins.png');
imshow(I)
```


(a)

```
%sobel 边界探测器
BW1 = edge(I,'sobel');
%canny 边界探测器
BW2 = edge(I,'canny');
figure
imshow(BW1)
figure, imshow(BW2)
```

运行结果如图 11-27 所示，图（a）为原始图像，图（b）为采用 sobel 探测器获得的结果，图（c）为采用 canny 边界探测器获得的结果。





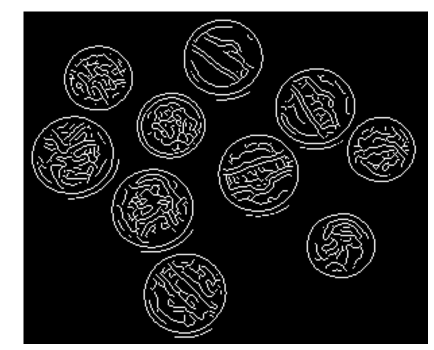


图 11-27　边界探测器在图像分析中的应用示例

11.5.3　图像调整

MATLAB 7.0 中的图像调整技术用于图像的改善。此处的改善有两个方面的含义，即客观方面，例如提高图像的信噪比；主观方面，例如通过修正图像的颜色和强度（灰度）使其某些特征更容易辨识。

1．灰度调整

在 MATLAB 7.0 中，相关的函数为 imadjust，该函数可用于调整灰度值或颜色图，下面

介绍其用法。

- J = imadjust(I,[low high],[bottom top],gamma)

将灰度图像 *I* 转换为图像 *J*，使值从 *low*～*high*，与从 *bottom*～*top* 相匹配。值大于 *high* 或小于 *low* 的被剪去，即小于 *low* 的值与 *bottom* 相匹配，大于 *high* 的值与 *top* 相匹配。使用该函数时可将[low high]或[bottom top]指定为空矩阵[]，此时缺省值为[0 1]。*gamma* 用来指定描述 *I* 和 *J* 值关系曲线的形状；gamma<1，越亮输出值越强；gamma>1，越亮输出值越弱；缺省值 gamma=1，表示线性变换。gamma 大于 1、等于 1 和小于 1 的映射方式如图 11-28 所示。

- newmap = imadjust(map,[low high],[bottom top],gamma)

对索引图像的颜色图进行变换。如果[low high]和[bottom top]均为 2×3 矩阵，则 gamma 为 1×3 向量，imadjust 函数分别调整红、绿、蓝成分，调整后的颜色图 *newmap* 大小与原来的 map 一样。

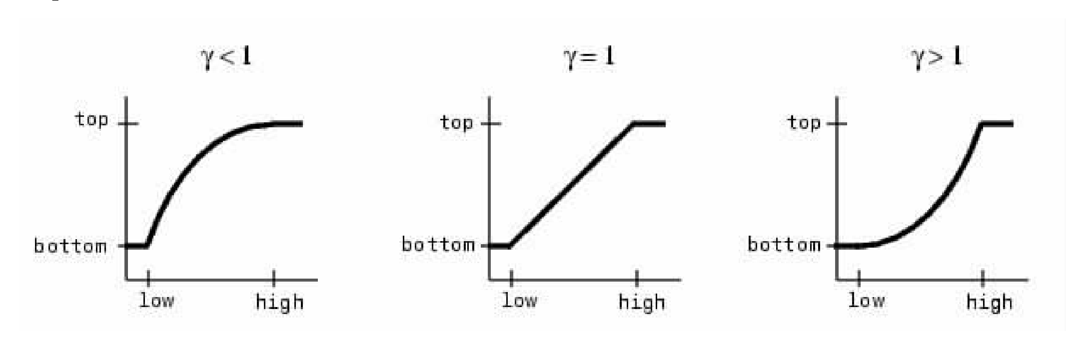


图 11-28　不同 gamma 对应的转移函数曲线

- RGB = imadjust(RGB1,...)

对于 RGB 图像 RGB1 的每个图像块进行调整。与调整颜色图一样，通过指定[low high]和[bottom top]均为 2×3 矩阵，gamma 为 1×3 向量，对每个图像块可以使用不同的参数值。如果 top<bottom，则图像颜色或灰度将倒置，即倒置变换，得到原图的底片。输入图像可以是 unit8 或双精度类型值，输出图像与输入图像值类型一致。

下面是一个调整图像的对比度的程序：

```
I = imread('cameraman.tif');
J = imadjust(I,[0 0.2],[0.5 1]);
subplot(1,2,1)
imshow(I)
subplot(1,2,2)
imshow(J)
```

程序执行结果如图 11-29 所示。




图 11-29　对比图调整前后的图像。

2．直方图调整法

在 MATLAB 7.0 中，histeq 函数用直方图均衡增强对比度。直方图均衡通过转换灰度图像亮度值或索引图像的颜色图值来增强图像对比度，输出图像的直方图近似与给定的直方图相匹配。

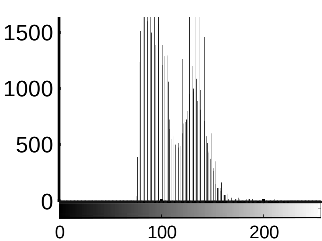
- J = histeq(I,hgram)：转换灰度图像 *I*，使输出图像 *J* 的直方图具有 length（hgram）个条，近似与 *hgram* 相匹配。向量 hgram 包含等间隔灰度值的整数计数个数；
- J = histeq(I,N)：将灰度图像 *I* 转换成具有 *N* 个离散灰度级的灰度图像 *J*，*N* 缺省值为 64；
- [J,T] = histeq(I)：返回灰度级变换，使 *J* 的灰度级与 *I* 的灰度级相匹配；
- newmap = histeq(X,map,hgram)：变换索引图像 *X* 的颜色图，使索引图像（X，newmap）的灰度级成分与 *hgram* 相匹配。返回变换后的颜色图 *newmap*，length(hgram)必须与 size(map,1)一样。

下面是对图像 pout.tif 进行直方图均匀化的程序：

```
%读入图像
I = imread('pout.tif');
%进行直方图均匀化
J = histeq(I);
subplot(2,2,1)
imshow(I);
subplot(2,2,2)
imshow(J)
subplot(2,2,3)
imhist(I)
subplot(2,2,4)
imhist(J)
```

程序运行结果如图 11-30 所示，上面两幅图分别为原始图像和经过直方图均匀化后的图像，下面两幅分别为它们所对应的直方图。





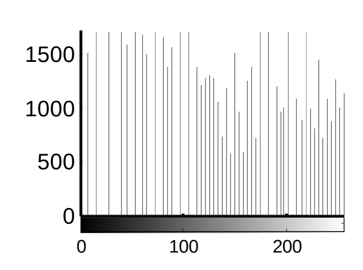


图 11-30　原始图像及直方图与直方图均匀化后的图像及直方图

11.5.4　图像平滑

图像平滑主要目的是减少图像噪声。图像噪声来自多方面，有来自于系统外部的干扰（如经过电源窜进系统内部的外部噪声），也有来自于系统内部的干扰（如摄像机的热噪声）。实际获得的图像都因受到干扰而含有噪声，噪声产生的原因决定了噪声分布的特性及与图像信号的关系。减少噪声的方法可以在空间域或在频率域处理，即：在空间域进行时，基本方法就是求像素的平均值或中值；在频率域中则运用低通滤波技术。

图像处理工具箱提供了许多方法来去除图像中的噪声，针对不同的噪声类型，有不同的去噪的方法。这些方法概括起来有以下三大类：

- 线性滤波；
- 中值滤波；
- 自适应滤波。

线性滤波的方法所采用的线性滤波器大部分在第 10 章中做了详细的介绍，所以这里我们着重对后面的两种处理方法进行介绍。

1．中值滤波

中值滤波是一种非线性信号处理方法，与其对应的中值滤波器也就是一种非线性滤波器。中值滤波器是在 1971 年由 J．W．Jukey 首先提出并应用在一维信号处理技术中（时间序列分析），后来应用到二维图像平滑中。在一定的条件下可以克服线性滤波器如最小均方滤波平均值滤波（平滑滤波）等所带来的图像细节模糊，而且对滤除脉冲干扰及图像扫描噪声最为有效。在实际运算中不需要图像的统计特性，这带来很多方便，但是对一些细节多，特别是点、线、尖顶部的图像不宜采用中值滤波方法。

在 MATLAB 7.0 图像处理工具中，提供了 medfilt2 函数用于实现中值滤波。

- B = medfilt2(A,[M N])：对矩阵 A 进行二维中值滤波。每个输出像素包含输入图像中相应像素周期的 $M\times N$ 临域的中值。在图像边缘添加 0，因此边缘在[M N]/2 内的点可能发生扭曲。[M N]缺省值为[3 3]；
- B = medfilt2(A,'indexed',…)：将 A 当作索引图像处理，如果 A 为 unit8 类，填补 0；如果 A 为双精度类则填补 1。

下面给出的例子将采用中值滤波对加有椒盐噪声的图像进行滤波，源代码设置如下：

```
%读入图像
I = imread('eight.tif');
imshow(I)
%加入椒盐噪声
J = imnoise(I,'salt & pepper',0.02);
%显示加入噪声后的图像
figure, imshow(J)
%进行中值滤波
L = medfilt2(J,[3 3]);
%显示中值滤波后的图像
figure, imshow(L)
```

程序运行结果如图 11-31 所示。

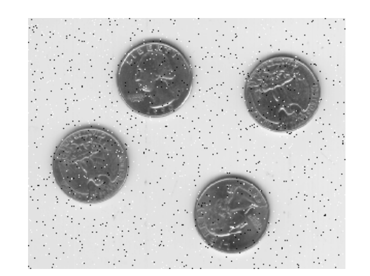




图 11-31　对图像进行中值滤波示例

2．自适应滤波

MATLAB 7.0 的图像处理工具箱中提供了 wiener2 函数来实现自适应滤波，该函数的用法如下：

- J = wiener(I,[M N],noise)：使用 $M\times N$ 大小领域局部图像均值与偏差，采用像素式自适

应滤波器对对象 I 进行滤波。

自适应滤波对高斯白噪声的去除效果最好，下面是自适应滤波的示例代码：

```
%载入图像
RGB = imread('saturn.png');
%RGB 转换为灰度图
I = rgb2gray(RGB);
%加入高斯白噪声
J = imnoise(I,'gaussian',0,0.005);
%采用自适应滤波
K = wiener2(J,[5 5]);
%显示原始图像、加入噪声的图像以及滤波后的噪声
imshow(I)
figure,imshow(J)
figure, imshow(K)
```

程序运行结果如图 11-32 所示，左上图为原始图像，右上图为加入高斯噪声后的图像，下面的图为经过滤波后的图像。

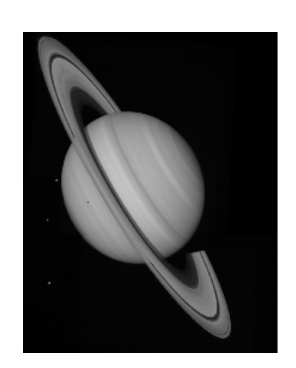
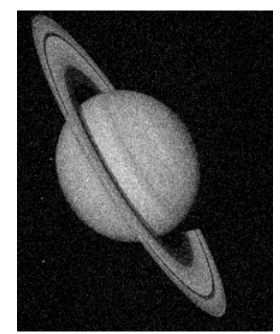
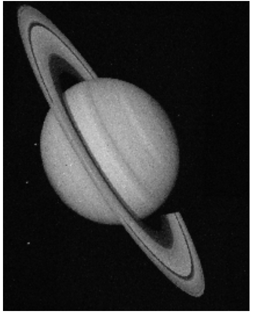




图 11-32　自适应滤波示例

11.6　特殊区域处理

11.6.1　区域的指定

在进行图像处理时，有时只要对图像中某个特定区域进行处理，并不需要对整个图像进行处理。MATLAB 7.0 中对特定区域的处理是通过二值掩模来实现的，通过选定一个区域后会生成一个与原图大小相同的二值图像，选定的区域为白色，其余部分为黑色。通过掩模图像，就可以实现对特定区域的选择性处理。下面介绍创建区域的方法：

1．多边形选择方法

roipoly 函数用于设定图像中的多边形区域，该函数返回与输入图像大小一致的二值图像 BW，选中的区域值为 1，其余的部分值为 0。其语法格式为：

- BW = roipoly(I,c,r)：用向量 *c*、*r* 指定多边形各角点的 *X*、*Y* 轴的坐标。
- BW = roipoly(I)：是让用户交互选择多边形区域，通过单击鼠标设定多边形区域的角点，用空格键和 Del 键撤销选择，按 Enter 键确认选择，确认后该函数返回与输入图像大小一致的二值图像 *BW*，在多边形区域内像素值为 1，其余区域内像素值为 0。

下面是一个根据指定的坐标选择一个六边形区域的程序：

```
%载入图像
I = imread('eight.tif');
%指定六边形的角点
c = [222 272 300 272 222 194];
r = [21 21 75 121 121 75];
%生成对应的二值图像
BW = roipoly(I,c,r);
figure,imshow(I);
figure,imshow(BW);
```

运行结果如图 11-33 所示。





图 11-33　根据指定的坐标选择六边形

2．其他选择方法

另外，在 MATLAB 7.0 的图像处理工具箱中提供了可以实现按灰度选择区域的函数 roicolor 函数，其语法格式为：

- BW = roicolor(A,low,high)：按指定的灰度范围分割图像，返回二值掩模 *BW*，[low high] 为所要选择区域的灰度范围，如果 *low* 大于 *high*，则返回空矩阵；
- BW = roicolor(A,v)：按向量 *v* 中指定的灰度值来选择区域。

下面是一个按灰度图像中的目标进行分割的程序：

```
%载入图像
I = imread('coins.png');
%选择图像灰度范围在 128～255 之间的像素
BW = roicolor(I,128,255);
figure,imshow(I)
figure,imshow(BW)
```

运行结果如图 11-34 所示。

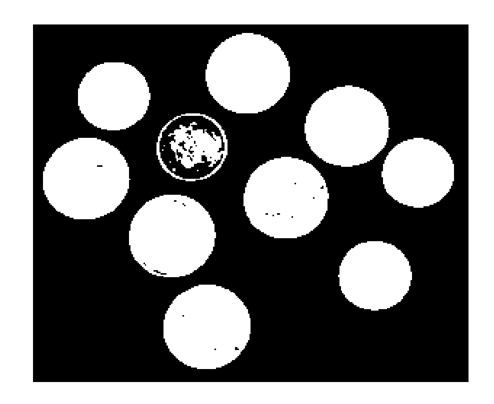



图 11-34　按照灰度范围选择图像区域示例

11.6.2　特定区域滤波

在 MATLAB 7.0 的图像处理工具箱中提供的 roifilt2 函数用于对特定区域进行滤波，其语法格式为：

- J = roifilt2(h,I,BW)：使用滤波器 *h* 对图像 *I* 中用二值掩模 *BW* 选中的区域滤波；
- J = roifilt2(I,BW,fun,P1,P2,…)：对图像 *I* 中用二值掩模 *BW* 选中的区域做函数运算 *fun*，其中 *fun* 是描述函数运算的字符串，参数为 *P1*、*P2*、…。返回图像 *J* 在选中区域的像素为图像 *I* 经 *fun* 运算的结果，其余部分的像素值为 *I* 的原始值。

下面是一个对指定区域进行锐化滤波的程序清单：

```
%载入图像
I = imread('pout.tif');
```

```
%生成对应的二值图像
imshow(I)
%交互式选择区域
BW = roipoly
%定义滤波器
h = fspecial('unsharp');
%进行区域滤波
I2 = roifilt2(h,I,BW);
figure, imshow(I2)
```

运行结果如图 11-35 所示。从图上可以看到，三角形中的图形发生了锐化，而其他部分没有变化。

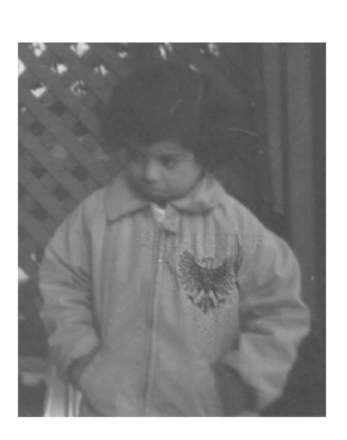



图 11-35　对特定区域进行滤波示例

11.6.3　特定区域填充

在 MATLAB 7.0 的图像处理工具箱中提供的 roifill 函数用于对特定区域进行填充，其语法格式为：

- J = roifill(I,c,r)：填充由向量 *c*、*r* 指定的多边形，*c* 和 *r* 分别为多边形各定点的 *x*、*y* 坐标。它是通过解边界的拉普拉斯方程，利用多边形边界的点的灰度平滑的插值得到多边形内部的点。通常可以利用对指定区域的填充来“擦”掉图像中的小块区域；
- J = roifill(I)：用户交互选取填充的区域。选择多边形的角点后，按 Enter 键确认选择，用空格键和 Del 键表示取消一个选择；
- J= roifill(I,BW)：用掩模图像 BW 选择区域。

下面是特定区域填充的示例代码。

```
%载入图像
load trees
%RGB 转换为灰度图
I = ind2gray(X,map);
```

```
imshow(I)
%指定特定区域，并进行填充
I2 = roifill;
imshow(I2)
```

运行结果如图 11-36 所示。

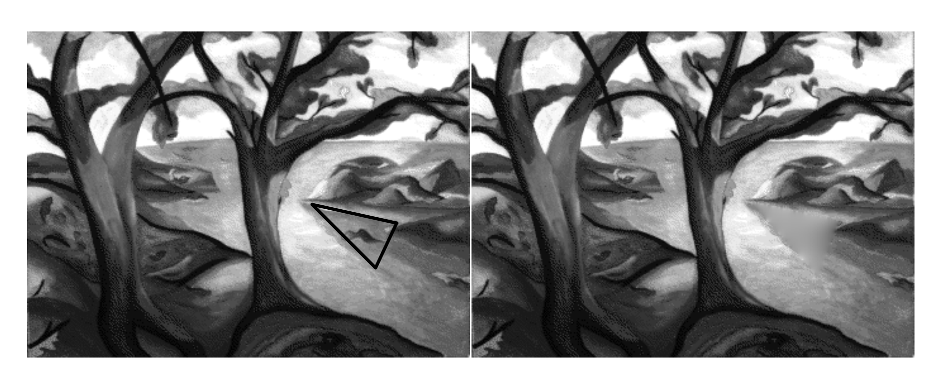


图 11-36　对指定区域进行填充示例

第 12 章　高级图形设计

通过第 5 章的学习，读者已经掌握了使用 MATLAB 7.0 的高级绘图函数和工具来创建一幅二维或三维图形的方法。而当用户想定制图形，并且要打算对图形的每个可能方面进行控制时，那么句柄图形会为用户提供强有力的工具。另外，在很多场合，用户需要编写一个可以多次反复使用的实用函数，这时候菜单、按钮、文本框作为输入方法就显得更有意义。所以，图形界面的实现对很多用户来讲是必不可少的一部分。

本章的前 3 节内容将系统地阐述句柄图形体系、图形对象、属性和操作方法，后面两节叙述图形用户界面地设计原则以及操作步骤，其中包括如何使用编程语言来实现 GUI。最后，给出一个实际的例子，帮助读者理解图形界面制作的整个过程。

12.1　句柄图形

句柄图形（Handle Graphics）是一种面向对象的绘图系统。该系统提供创建计算机图形所必须的各种软件。它所支持的指令，可以直接创建线、文字、网格、面以及图形用户界面。在第 4 章所介绍的各种 MATLAB 7.0 高层（High-level）图形指令（如 plot、mesh）都是以句柄图形软件为基础写成的。也正是这个原因，句柄图形也被称为低层（Low-level）图形。

12.1.1　图形对象、图像句柄和句柄图形树结构

1．图形对象

MATLAB 7.0 把用于数据可视和界面制作的基本绘图要素称为句柄图形对象（Handle graphics object）。构成 MATLAB 7.0 句柄图形体系的 12 个图形对象如图 12-1 所示。每个图形对象可以被独立地操作。

在 MATLAB 7.0 中生成的每个具体图形，由若干不同对象构成。每个具体图形不必包含全部对象，但每个图形必须具备根屏幕和图形窗（简称图）。

2．句柄

每个具体对象都有一个惟一的身份（Identifier），即句柄（Handle）。句柄是存取图形对象惟一规范的识别符。不同对象的句柄不可能重复和混淆。

每台计算机的根对象只有一个，即屏幕。它的句柄总是数字 0。而简称为图的图形窗（Figure Windows）的句柄总是正整数，它用来标识图形窗的序号。除了以上两种对象外，其余对象的句柄都是双精度浮点数。

注意：对根屏幕、图对象来说，数字可直接作为调用对象的句柄。但不要企图通过直接输入浮点

数来作为其他对象的句柄；这些对象的句柄只能由相关指令运作得到。

3．句柄图形的结构

在句柄图形体系中，各图形对象并非平等，它们之间的关系如图 12-1 所示。

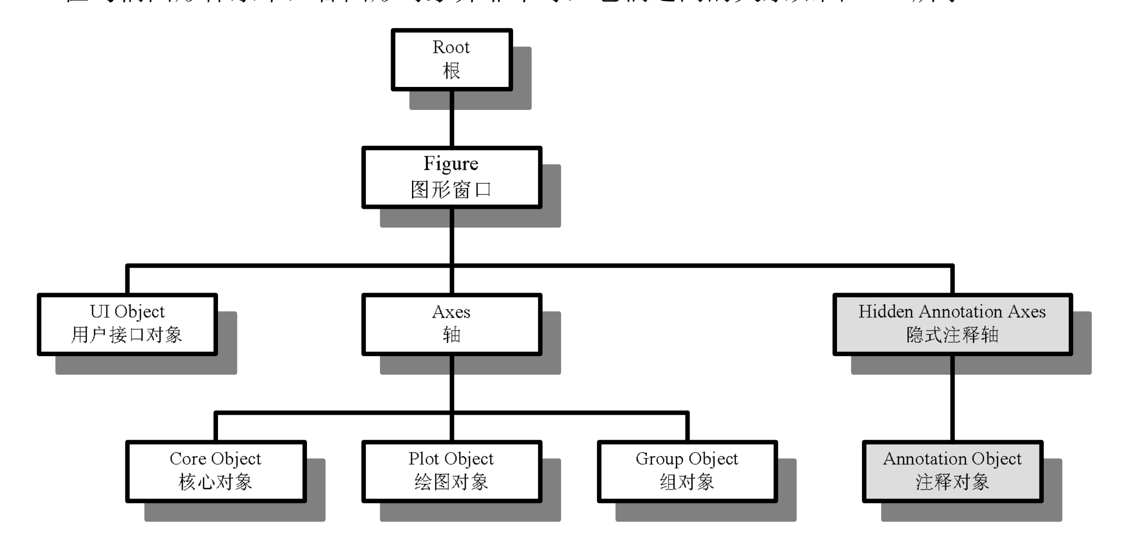


图 12-1　句柄图形体系结构图

- 处于树结构最高层的图形对象是根，它是所有其他图形对象的“父（Parent）”。图对象是根对象的直接“子”对象。理论上，一个根屏幕的独立图形窗数量不限。
- 图形窗有 3 各不同类型的“子”，即用户接口对象、轴以及隐式的注释轴。
- 轴有 3 种不同类别的“子”，即核心对象、绘图对象以及组对象。而注释轴有一个“子”，即注释对象。

12.1.2　图形对象种类

在 MATLAB 7.0 中，有两种基本的图形对象类型。

- 核心图形对象（Core graphics objects）：用于实现高层图形指令，是组合对象的基本元素，示例图形如图 12-2 所示；
- 组合对象（Composite objects）：由核心图形对象组成，用来实现特殊的绘图功能。用来构成图形对象的 3 种子对象。

下面分别介绍图 12-1 中示的几种低层的对象。

- 绘图对象：由基本图形对象组成，支持在本层次上进行对象属性的修改；
- 注释对象：存在于不同于其他任何图形对象的层上；
- 组对象：生成一组对象，它们可以完成某种特殊的功能，但在外面看来象单一对象一样；
- 用户接口对象：主要用来构建用户图形界面接口。

根对象（root）：位于体系结构图中的最顶端是 root 根对象。根对象相当于计算机的显示屏。一个 GUI 中仅有一个根对象，其他所有对象都是它的“子”。用户是不能创建根对象的，

当用户启动 MATLAB 7.0 时根对象就已经存在。用户通过设置 root 的属性值改变图形的显示效果。

图形窗口（figure）：figure 对象是位于 root 显示屏上的单独窗口，MATLAB 7.0 在这个窗口中显示图形。MATLAB 7.0 对 figure 对象的数量没有限制，用户以创建任意多个图形窗口。所有的 figure 对象都是 root 的子对象，而除了 root 以外的所有其他对象都是 figure 的子对象。如果调用绘图函数时图形窗口不存在，所有的绘图函数都能够自动创建一个 figure 对象。如果在 root 中有许多个 figure 对象，其中有一个将被定义为“当前”窗口作为图形输出的目标。

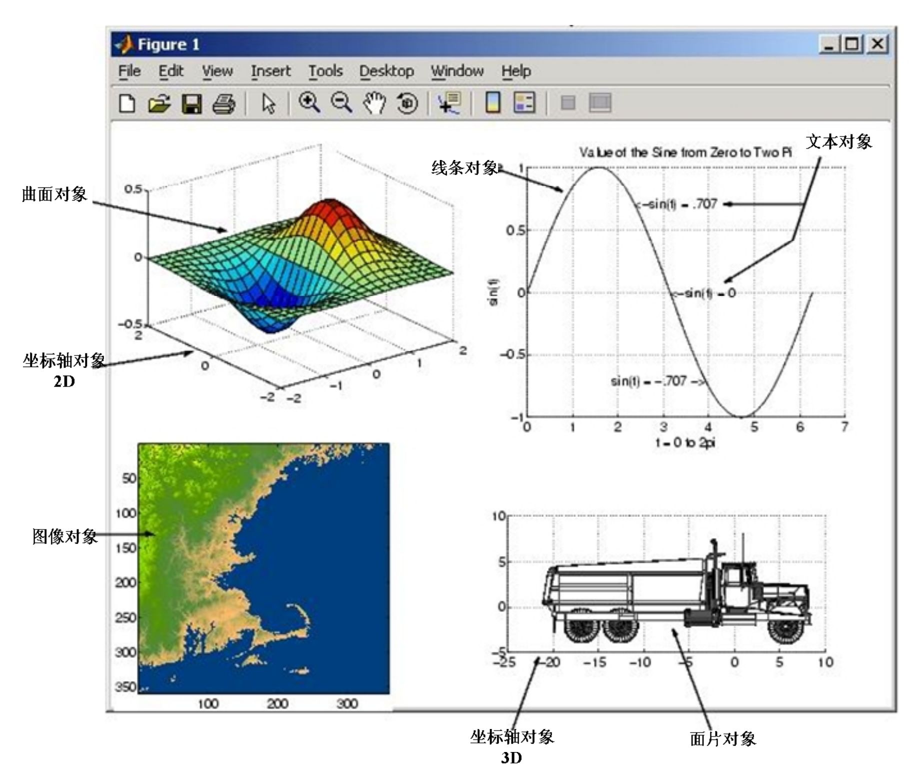


图 12-2　核心图形对象示例

核心图形对象（core object）：包含了最基本的绘图元素，例如坐标轴对象、图像对象、灯光对象、线条对象、面片对象、矩形对象、曲面对象以及文本对象。

坐标轴对象在图形窗口中定义了一个区域并使用这个区域为自己的子对象定向。坐标轴对象是 figure 的子对象，也是图像对象、灯光对象、线条对象、面片对象、矩形对象、曲面对象以及文本对象的父对象。如果创建图形时不存在坐标轴对象，那么所有的绘图函数将会自动创建一个坐标轴对象；如果存在多个坐标轴对象，则指定其中一个为“当前”坐标轴对象。

图像对象是由一个数据矩阵和一个颜色映射矩阵（可不存在）组成。图像可以分成 3 种类型，即真彩、索引和灰度，不同图像类型之间的区别在于数据矩阵和像素颜色的关系不同。由于图像是一个严格的二维图像，因而用户仅可以在二维视图中观察图像。

灯光对象定义光源，该光源对坐标轴中所有的面片对象和曲面对象产生影响。虽然用户

看不到光源，但是可以通过设置光源的属性来控制光源。

线条对象是创建大多数二维和三维绘图的基本图形元素。高级函数 plot、plot3 和 loglog 等可以用来创建线条对象。坐标轴对象的坐标系统决定了线条对象的位置和方向。

面片对象由有边缘的多边形构成。单个的面片可以有许多个小面，每个小面可以独立地使用均匀或插值方法着色。

矩形对象是一个二维填充区域，其形状可以是矩形或椭圆。

曲面对象是矩阵数据的三维描述图形，使用矩阵元素作为点到 x-y 平面的高度，通过调用绘图函数绘制而成。曲面图形由多个四边形组成，四边形到顶点由矩阵数据决定。

文本对象即字符串，由坐标轴对象的坐标系统来定义字符串的位置。在图 12-2 中展示了各种核心图形对象。

12.1.3 图形对象属性概念

通过修改图形对象的属性可以控制对象外观、行为等许多特征。属性不但包括对象的一般信息，而且包括特殊类型对象独一无二的信息。例如，用户可以从任意给出的 figure 对象中获得以下信息：窗口中最后一次输入的标识符、指针的位置以及最近一次选择的菜单项。MATLAB 7.0 将所有图形信息组织在一个层次表中，并将这些信息存储在相应的属性中。例如，root 属性表包括当前图形窗口的句柄和当前的指针位置；figure 属性包括其子对象的类型列表，同时实时跟踪窗口中发生的事件；axes 属性包含有关其子对象对图形窗口映射表的使用方式以及 plot 函数所使用的颜色命令。

用户不但可以查询当前任意对象的任意属性值，而且可以指定大多数属性的取值（某些属性为 MATLAB 7.0 控制的只读属性）。属性值仅对对象的特定实例起作用，也就是说，修改属性值不会对同类对象、不同实例的属性产生影响。可以通过设置属性的默认值来影响所有此后创建的对象的属性。如果用户既没有定义默认值也没有在创建对象时指定属性值，MATLAB 7.0 将使用系统默认值。每个对象创建函数的参考入口都提供一个与图形对象有关的属性完整列表。

有些属性是所有图形对象都具备的，例如类型（Type）、被选状态（Selected）、是否可见（Visible）和创建回调函数（CreateFcn）、销毁回调函数（DeleteFcn）。而有些属性则是某种对象独有的，例如线条对象的线性属性等。这些独有的属性将在介绍属性设置方法时具体介绍。

在不引起混淆的前提下，编程时允许使用属性名的缩写。但是，在编写 M 文件时最好不使用缩写，以防将来 MATLAB 7.0 系统扩展时导致属性重名现象。

set 和 get 函数可以指定或获得已存在对象的属性值，如果该属性值有一个取值范围集，这两个函数还能够将该属性所有可能的取值列举出来。设置已存在对象属性的基本语法格式如下：

```
set(object_handle,'PropertyName','NewPropertyValue')
```

如果希望查询指定对象的当前属性值，可以使用以下语句：

```
returned_value = get(object_handle,'PropertyName');
```

12.2　图形对象的操作

12.2.1　创建图形对象

典型的图形通常包括许多种相关的图形对象，由这些对象共同生成有具体含义的图形或图片。每一种类型的图形对象都有一个相对应的创建函数（除 root 对象外），这个创建函数使用户能够创建该对象的一个实例。对象创建函数名与所创建的对象名相同，例如，函数 text 将创建一个 text 对象，figure 函数将创建一个 figure 对象。表 12-1 列出了 MATLAB 7.0 所有的对象创建函数。

表 12-1　　图形对象创建函数及其创建对象描述

函数	对象描述
Axes	标度和定向 axes 子对象 image、light 等的矩阵坐标系统
Figure	显示图形的窗口
Image	使用颜色映射表索引或 RGB 值的二维图片。数据可以是 8 位也可以是双精度数据
Light	位于坐标轴中，能够影响面片和曲面的有方向光源
Line	由顺序链接坐标数据的直线段构成的线条
Patch	将矩阵的每一列理解为由一个多边形构成的小面
Rectangle	矩形或椭圆形的二维填充区域
Surface	由矩阵数据（理解为高度）定义的矩形创建而成的曲面
Text	位于坐标轴系统中的字符串
Uicontextmenu	与其他图形对象相关的用户文本菜单
Uicontrol	可编程用户接口空间，例如按钮、滚动条和列表框等
uimenu	在图形窗口顶端出现的菜单

所有的对象创建函数都有相同的调用格式：

```
返回句柄＝创建函数名（'属性名'，属性值，…）
```

用户可以使用属性名和属性值参数对为任意的对象属性指定一个数值（除了只读类型属性以外）。函数将返回创建对象的句柄，在这之后用户可以使用这个句柄查询或修改所创建对象的属性值。下面给出一个创建对象的实例，首先对一个数学函数求值，再使用 figure、axes 和 surface 函数创建 3 个图形对象并设置其属性，其他图形对象的属性使用 MATLAB 7.0 默认值。

源程序代码如下：

```
%设置 x,y 的值
[x,y] = meshgrid([-2:.4:2]);
%生成 z 的值
Z = x.*exp(-x.^2-y.^2);
%创建图形对象
fh = figure('Position',[350 275 400 300],'Color','w');
ah = axes('Color',[.8 .8 .8],'XTick',[-2 -1 0 1 2],...
```

```
                    'YTick',[-2 -1 0 1 2]);
sh = surface('XData',x,'YData',y,'ZData',Z,...
                    'FaceColor',get(ah,'Color')+.1,...
                    'EdgeColor','k','Marker','o',...
                    'MarkerFaceColor',[.5 1 .85]);
```

注意：surface 函数并不是像 surf 函数那样采用三维视图。

程序运行结果如图 12-3 所示。

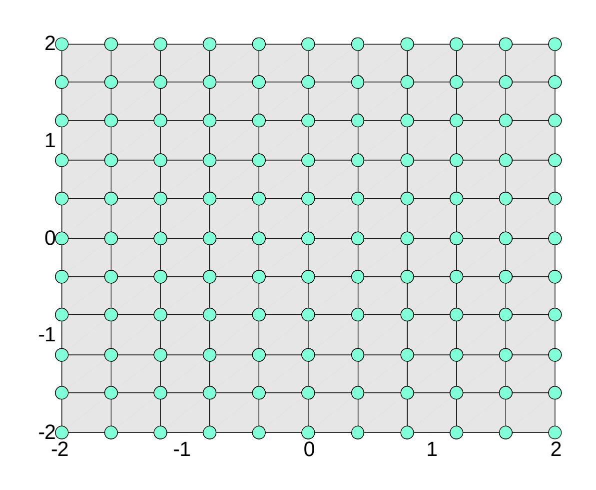


图 12-3　创建图形对象实例

为了清楚地观察到创建的各个图形对象，可以通过镜头命令或 view 命令改变视角：

```
view(3)
```

改变视角后的图形外观如图 12-4 所示。

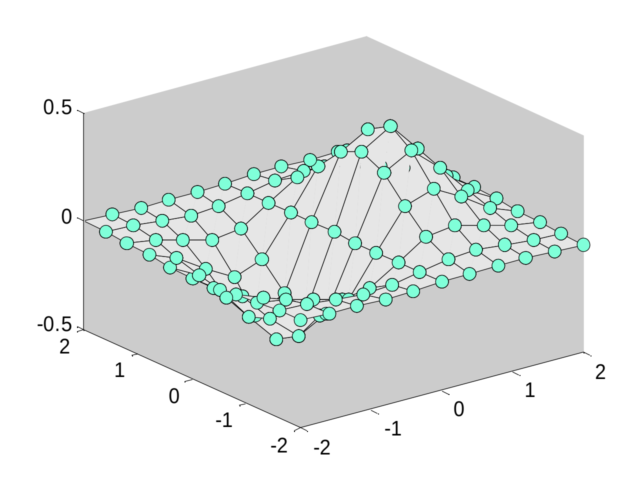


图 12-4　视图改变后的图形效果

对象创建函数还能够使用一种较为简单的调用形式，例如下面的示例代码：

```
text(0.5,0.5,0.5,'Hello');
```

或者采用如下代码

```
text('Position',[0.5,0.5,0.5],'String','Hello')
```

上述两种方式是等效的。

> 注意：使用对象创建函数的这种形式有时会导致输出效果与正常调用形式的结果有细微的不同。

在默认情况下，所有的创建函数都将当前的图形窗口和坐标轴作为其创建对象的父对象。用户也可以指定所创建对象的父对象。例如下面的示例代码：

```
axes('Parent',figure_handle,…)
```

上述代码将在 figure_handle 指定的图形窗口中创建一个坐标轴。用户可以通过定义 parent 属性将创建的对象从一个父对象中转移到另一个父对象中，示例代码设置如下：

```
set(gca,'Parent',figure_handle,…)
```

实际上，MATLAB 7.0 高级绘图函数（例如 plot、surf 等）也是通过调用相应的创建函数来绘制图形对象的。高级函数可能会根据坐标轴和图形窗口的 NextPlot 属性来判断是否清除坐标轴或创建新的图形窗口；而低级对象创建函数则不考虑图形窗口和坐标轴的 NextPlot 属性，只是创建各自的图形对象并将这些对象放置在当前的父窗口中。例如，第一次调用下面的代码：

```
line('XData',x,'YData',y,'ZData',z,'Color','r')
```

此时，将使用指定的数值在当前坐标轴中绘制一条红线（如果不存在坐标轴，MATLAB 7.0 将创建一个坐标轴：如果不存在图形窗口，MATLAB 7.0 也将创建一个图形窗口）。如果用户是第二次调用 line 函数，MATLAB 7.0 将在当前坐标轴中绘制第二条直线，但并不擦除第一条直线。高级绘图函数（例如 plot）则与之不同，它们将擦除以前的图形对象并重置所有的坐标轴（除了位置和比例尺）。用户通过使用 hold 命令或改变 NextPlot 属性来改变高级绘图函数的这种行为。

12.2.2　图形对象属性设置

在创建图形对象的同时，用户可以根据自己的需要来设置相应图形对象的属性。用户还可以通过 set 函数和创建函数返回的句柄来改变一个已存在对象的属性。例如，将坐标轴的 *y* 轴移动到当前图形的右边，示例代码如下：

```
set(gca,'YAxisLocation','right')
```

如果句柄参数是一个向量，MATLAB 7.0 将对所有由该向量指定的对象进行属性设置。

可以使用结构体数组或单元数组来定义属性名并设置属性值，这在希望对某些对象使用相同的属性值时非常有用。例如，用户可以定义一个结构体来设置坐标轴的属性，使之符合特定图形的显示要求，示例代码设置如下：

```
view1.CameraViewAngleMode = 'manual';
view1.DataAspectRatio = [1 1 1];
view1.ProjectionType = 'Perspective';
```

为了设置当前坐标轴的这些属性值，可以输入如下语句：

```
set(gca,view1)
```

还可以使用 set 函数来显示多个属性的可能取值但不真正设置新的属性值。例如，以下

语句将获得 line 对象属性的可能取值：

```
set(obj_handle,'Marker')
```

MATLAB 7.0 将返回由 obj_handle 定义的 Marker 属性的数值列表：

```
[ + | o | * | . | x | square | diamond | v | ^ | > | < | pentagram | hexagram | {none} ]
```

如果希望得到所有可设置的属性，或者希望得到能够接收字符串属性的所有可能取值，使用对象句柄作为 set 函数的惟一参数即可实现：

```
set(object_handle)
```

例如，对于一个曲面对象调用以上语句，MATLAB 7.0 将返回如下信息：

```
CData
CDataScaling: [ {on} | off]
EdgeColor: [ none | {flat} | interp ] ColorSpec.
EraseMode: [ {normal} | background | xor | none ]
FaceColor: [ none | {flat} | interp | texturemap ] ColorSpec.
LineStyle: [ {-} | -- | : | -. | none ]
Visible: [ {on} | off ]
```

set 函数的输出是一个结构体数组。例如下面的示例代码：

```
a = set(gca);
```

上述语句中的返回值 *a* 是一个结构体，其域名是对象的属性名，域值是属性的可能取值。如果需要获得坐标轴网线格式的可能取值，可以使用以下的语句：

```
a.GridLineStyle
```

MATLAB 7.0 将输出的代码如下：

```
ans =

    '-'
    '--'
    ':'
    '-.'
    'none'
```

> 注意：虽然属性名与大小写无关，但结构体域名与大小写有关，如果输入：
>
> a.GridLineStyle
>
> MATLAB 7.0 将产生一个错误信息。

12.2.3 属性值查询

使用 get 函数来查询一个或多个对象属性的当前取值。例如，使用以下语句检查当前坐标轴 PlotBoxAspectRatio 属性的取值：

```
get(gca,'PlotBoxAspectRatio')
```

执行结果代码如下：

```
ans =
     1     1     1
```

虽然 MATLAB 7.0 能够列举所有属性的取值，但对于包含数据的属性，例如 Current 和 ColorOrder，MATLAB 7.0 仅列举该属性值的维数。例如下面的示例代码：

```
AmbientLightColor = [1 1 1]
Box = off
CameraPosition = [0.5 0.5 2.23205]
CameraPositionMode = auto
CameraTarget = [0.5 0.5 0.5]
CameraTargetMode = auto
CameraUpVector = [0 1 0]
CameraUpVectorMode = auto
CameraViewAngle = [32.2042]
CameraViewAngleMode = auto
CLim: [0 1]
CLimMode: auto
Color: [0 0 0]
CurrentPoint: [ 2x3   double]
ColorOrder: [ 7x3   double].
Visible = on
```

用户可以使用指定的属性名单独查询某一属性，从而获得该属性的具体数据：

```
get(gca,'ColorOrder')
```

得到的属性为：

```
ans =
         0         0    1.0000
         0    0.5000         0
    1.0000         0         0
         0    0.7500    0.7500
    0.7500         0    0.7500
    0.7500    0.7500         0
    0.2500    0.2500    0.2500
```

如果用户将输出指定的某一变量，则 MATLAB 7.0 将该变量作为一个结构体数组进行赋值，该数组的域名是对象的属性名，域值是属性名对应的当前取值。例如，如果用户希望绘制数据 x 和 y，可以执行下面的指令：

```
h = plot(x,y);
```

还可以通过下面的指令获得由 plot 函数创建的 line 对象的所有属性：

```
a = get(h);
```

下面用户就可以通过使用 *a* 的域名来访问 line 属性的数值。以下语句调用 text 函数将字符串“x and y”作为数据 *x* 和 *y* 的起点，文本使用与 line 相匹配的颜色：

```
text(x(1),y(1),'x and y data','Color',a.Color)
```

如果 *x* 和 *y* 是矩阵，plot 函数将为矩阵的每一列绘制一条线。为了标记数据第二列产生的线条，使用以下语句：

```
text(x(1,2),y(1,2),'Second set of data','Color',a(2).Color)
```

如果希望查询属性群，可以定义一个属性名单元数组并使用它方便地获取这些属性取值。例如，假设用户希望查询坐标轴 camera mode 的属性，则定义以下的单元数组：

```
camera_props(1) = {'CameraPositionMode'};
camera_props(2) = {'CameraTargetMode'};
camera_props(3) = {'CameraUpVectorMode'};
camera_props(4) = {'CameraViewAngleMode'};
```

使用这个单元数组作为参数来获得这些属性的当前取值：

```
get(gca,camera_props)
```

运行结果代码如下：

```
ans =
    'auto' 'auto' 'auto' 'auto'
```

12.2.4 设置用户属性默认值

如果用户没有指定属性值，同时又禁止使用默认的属性值，那么 MATLAB 7.0 将使用系统默认属性值为所有对象定义属性值。用户可以使用以下语句获得所有属性的系统默认属性值：

```
a = get(0,'Factory');
```

get 函数将返回一个结构体数组，该数组的域名由对象类型和属性名联合构成，域值是域名所标识的对象和属性的系统默认值。例如下面的示例代码：

```
UimenuSelectionHighlight: 'on'
```

标识 uimenu 对象 SelectionHighlight 属性的系统默认属性值为 on。可以使用以下语句获得单个属性的系统默认属性值：

```
get(0,'FactoryObjectTypePropertyName')
```

例如下面的示例代码：

```
get(0,'FactoryTextFontName')
```

对于所有对象的属性，MATLAB 7.0 系统中都对应有一个默认的属性值。

在进行绘图或其他需要了解图像属性值的工作时，MATLAB 7.0 将从当前对象开始，在继承表中始终向上搜索，直到找到用户定义的默认值或系统，因此总能找到合适的属性值。可以看出，用户定义的默认值越靠近继承表的 root 对象，MATLAB 7.0 的搜索范围越广。如果用户在 root 级定义 line 对象的默认值，MATLAB 7.0 将对所有的 line 对象使用这个默认值，如果用户仅在 axes 级定义 line 对象的默认值，MATLAB 7.0 将对所有的 line 对象使用这个默认值。如果用户在多级中定义对象的默认值，与该对象最近的父对象级中定义的默认值将被使用，因为该默认值是 MATLAB 7.0 最先找到的默认值。

这里需要注意的是默认值的设置仅对那些设置完成后所创建的对象有效，已存在的图形对象不会发生变化。指定默认值时首先要创建一个以 Default 开头、然后紧跟对象类型、最后是对象属性的字符串。例如，如果希望在当前的图形窗口级指定 line 对象的 LineWidth（线

的宽度）属性为 1.5 个点宽，可以使用以下语句：

```
set(gcf,'DefaultLineLineWidth',1.5)
```

字符串 DefaultLineLineWidth 将指明该属性是 line 对象的 LineWidth 属性。定义图形窗口的颜色可以使用字符串 DefaultFigureColor。这里要注意的是，仅在 root 级指定图形窗口的属性才是有意义的。

```
get(gcf,'default')
```

使用参数 default 可以将一个属性设置为 MATLAB 7.0 第一个搜索到的默认值。例如，以下语句将产生一个绿色的曲面 EdgeColor 属性（边缘属性）：

```
set(0,'DefaultSurfaceEdgeColor','k')
h = surface(peaks);
set(gcf,'DefaultSurfaceEdgeColor','g')
set(h,'EdgeColor','default')
```

程序执行结果如图 12-5 所示。

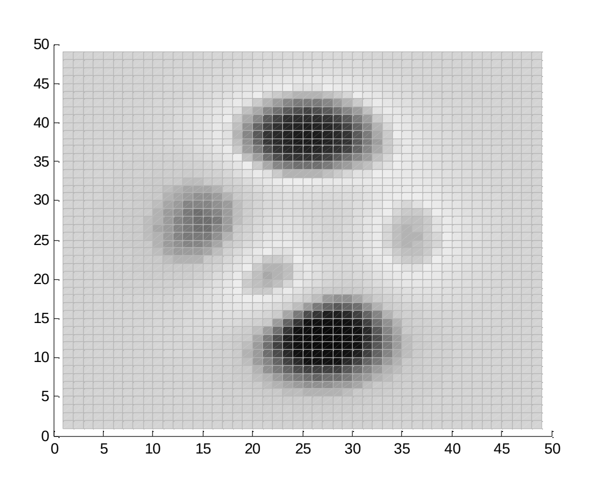


图 12-5　设置图形属性的默认值

在 figure 级存在一个该属性的默认值，MATLAB 7.0 将第一个找到这个默认值，于是用它来替换 root 级定义的默认值。

指定某个属性的数值为“remove”时将会删除用户定义的默认值：

```
set(0,'DefaultSurfaceEdgeColor','remove')
```

以上语句将 root 级定义的曲面 EdgeColor 默认值删除。

指定一个属性取值为“factory”，则该属性将采用系统默认的属性值。例如，以下语句将曲面的 EdgeColor 属性设置为系统默认属性值颜色——黑色，而用户设置的默认值将被忽略：

```
set(gcf,'DefaultSurfaceEdgeColor','g')
h = surface(peaks);
set(h,'EdgeColor','factory')
```

> 注意：在使用对象创建函数时，default、remove 和 factory 必须以反斜线开头说明该字符串为保留字。

下面给出两个例子来说明以上内容。plot 函数在显示多个图形时将循环使用由坐标轴的 ColorOrder 属性定义的颜色。如果用户为坐标轴的 LineStyleOrder 属性定义多个属性值，那么 MATLAB 7.0 将在每一次颜色循环后改变线性的宽度。用户还可以通过定义属性默认值使 plot 函数使用不同的线性来创建图形。下面给出几个通过修改属性值进行绘图的例子。

（1）创建一个白底色图形窗口，然后在 root 级为坐标轴对象设置默认值。

```
%创建一个白底色的窗口
whitebg('w')
%设置默认值
set(0,'DefaultAxesColorOrder',[0 0 0],...
        'DefaultAxesLineStyleOrder','-|--|:|-.')
Z = peaks; plot(1:49,Z(4:7,:))
```

无论何时用户调用 plot 函数，该函数将采用同一种颜色绘制所有数据，这是因为坐标轴的 ColorOrder 属性仅包括一个颜色。但是函数将循环使用 LineStyleOrder 定义的不同线型。程序运行结果如图 12-6 所示。

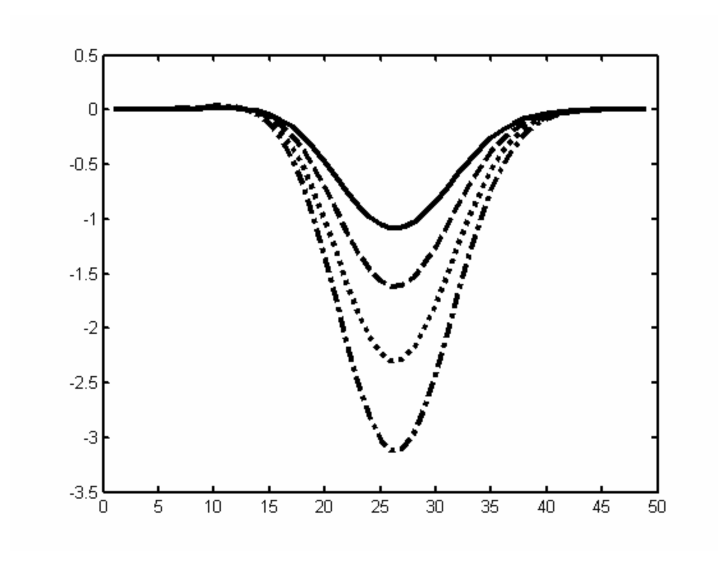


图 12-6　root 级设置对象默认值示例

（2）在继承表的多级上设置默认值。

在一个图形窗口中创建两个坐标轴，分别在 figure 级和 axes 级设置坐标轴颜色、线型等属性的默认值，示例代码设置如下：

```
t = 0:pi/20:2*pi;
s = sin(t);
c = cos(t);
%设置坐标轴颜色属性默认值
figh = figure('Position',[30 100 800 350],...
                'DefaultAxesColor',[.8 .8 .8]);

axh1 = subplot(1,2,1); grid on
```

```
%在第一个坐标轴处设置线型默认值
set(axh1,'DefaultLineLineStyle','-.')
line('XData',t,'YData',s)
line('XData',t,'YData',c)
text('Position',[3 .4],'String','Sine')
text('Position',[2 -.3],'String','Cosine',...
        'HorizontalAlignment','right')
axh2 = subplot(1,2,2); grid on
%在第二个坐标轴处设置文本旋转属性的默认值
set(axh2,'DefaultTextRotation',90)
line('XData',t,'YData',s)
line('XData',t,'YData',c)
text('Position',[3 .4],'String','Sine')
text('Position',[2 -.3],'String','Cosine',...
        'HorizontalAlignment','right')
```

此时，不同的子图形使用相同的 text 和 line 函数会得到不同的显示结果，如图 12-7 所示。

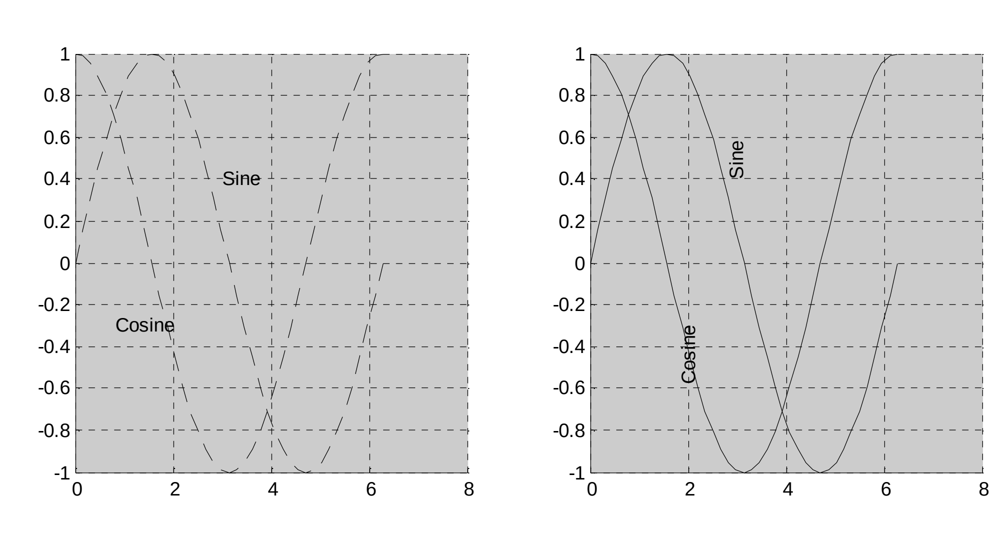


图 12-7　使用不同默认值属性值的不同结果

由于默认的坐标颜色（Color）属性是在图形窗口级（figure）设置的，所以 MATLAB 7.0 使用相同的灰色背景颜色创建坐标轴。作图的坐标轴点划线是默认线型，在该图中每一次调用 line 函数都使用这种线型，而右边的坐标轴没有定义默认线型，所以 MATLAB 7.0 使用系统默认属性值——实线型。右边的坐标轴给文本的 Rotation 属性定义为 90^0。该图中所有的文本都使用这种旋转角度，而左边的坐标轴给文本使用固有默认值，不旋转文本。

如果用户希望无论何时运行 MATLAB 7.0 都使用用户定义的默认值，可以在 startup.m 文件中指定这些默认值。

注意：在调用 colordef 命令时，MATLAB 7.0 可能会自动装载一些外观属性的系统默认值。

12.3 句柄使用方法

12.3.1 访问对象句柄

MATLAB 7.0 给所创建的每一个图形对象指定一个句柄。所有的对象创建函数都能够返回被创建对象的句柄。如果用户希望访问对象的属性，那么最好在创建对象时将对象的句柄赋给一个变量，以免以后对句柄进行重复搜索。当然，用户也可以使用 findobj 函数或通过查询其父对象的 Children 属性获得已存在对象的句柄。

除了 root 对象和 figure 对象以外，所有图形对象的句柄都是一个浮点数。用户使用这些句柄数据时必须保证这些数据的全部精度。root 对象的句柄总是 0；figure 对象的句柄可以是显示在窗口标题栏中的整数（默认情况下），也可以是一个完全符合 MATLAB 7.0 内部浮点数精度的数值。

图像对象的所有操作都是针对当前对象而言的。“当前”是图形对象的一个重要概念，当前窗口是指被指定作为图形输出的窗口；当前坐标轴是创建坐标轴子对象命令的目的坐标轴；当前对象是用户创建的最后一个图形对象或鼠标单击的最后一个对象。

MATLAB 7.0 分别将这些当前对象的句柄保存在其父对象的属性中，如图 12-8 所示。

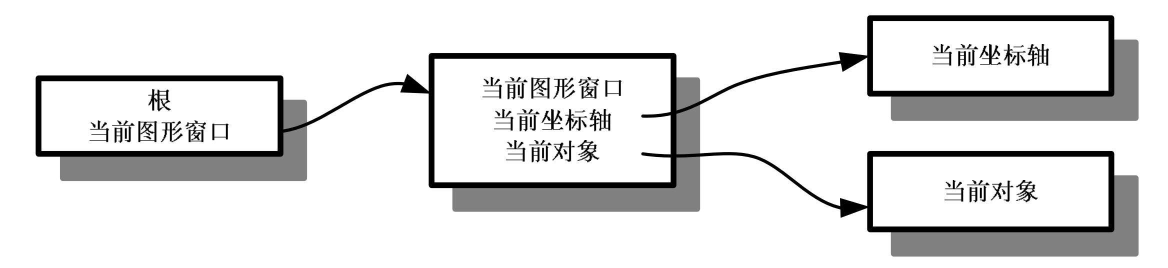


图 12-8　当前对象句柄保存途径

从图 12-8 中列出的属性可以获得这些关键对象的句柄，代码设置如下：

```
get(0,'CurrentFigure');
get(gcf,'CurrentAxes');
get(gcf,'CurrentObject');
```

以上语句中的 gcf、gca、gco 是 get 函数的速记符，它们的含义如下：

- gcf：返回 root 的 CurrentFigure 属性值；
- gca：返回当前图形窗口的 CurrentAxes 属性值；
- gco：返回当前图形窗口的 CurrentObject 属性值。

用户可以使用这些速记符（也可以称为命令）作为函数的对象句柄参数。例如，用户可以选择一个 line 对象，然后使用 gco 作为 set 函数的参数来指定该 line 对象的句柄：

```
set(gco,'Marker','square')
```

或者使用以下语句列举出所有当前坐标轴的属性值：

```
get(gca)
```

下面的语句可以获得当前坐标轴中的图形对象的句柄：

```
h = get(gca,'Children');
```

通过以下语句确定对象的类型：

```
get(h,'type')
```

最后执行结果的代码如下：

```
ans =
        'text'
        'patch'
        'surface'
        'line'
```

虽然 gcf 和 gca 提供了一个简单地获取当前窗口和坐标轴句柄的方法，但是很少在 M 文件中使用这两个命令，因为一般设计 MATLAB 7.0 程序 M 文件时，不会根据用户行为来获得当前对象。

MATLAB 7.0 还提供了一种通过属性搜索对象的方法，即 findobj 函数。findobj 函数能够快速地形成一个继承表的横截面并获得具有指定属性值的对象句柄。如果用户没有指定一个开始搜索的对象，findobj 函数将从 root 对象开始，始终搜索与用户指定属性名和属性值相符的所有事件。下面举例说明该函数的使用方法。

在如图 12-9 所示的 sin 函数图形中包括用来标注特殊点的文本对象。

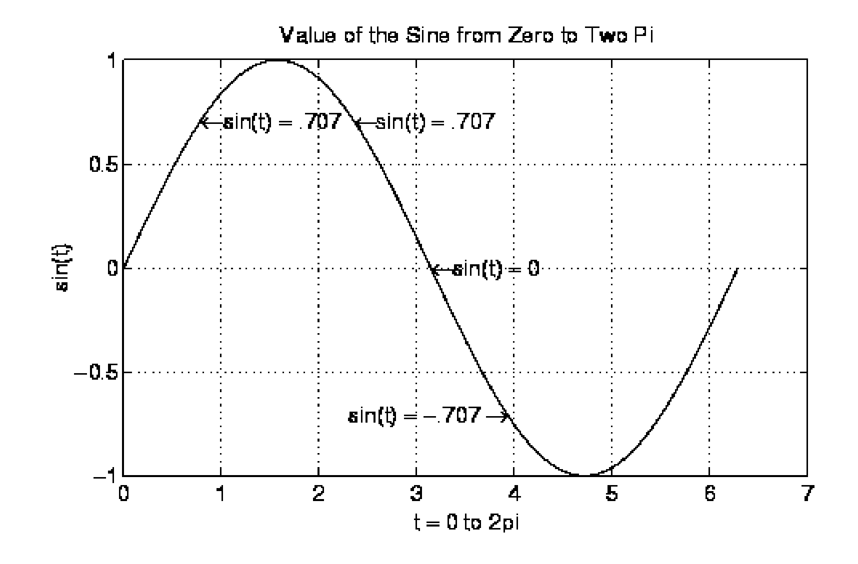


图 12-9　含有文本对象的图形

假设用户希望将文本字符串 sin（t）= 0.707 从当前位置移动到[3*pi/4,sin(3*pi/4)]。首先要获得文本对象的句柄，然后通过使用这个句柄修改文本对象的属性 Position。

为了使用函数 findobj，选择惟一一个能够表示该文本对象的属性值来搜索当前对象的句柄。在本例中使用文本对象的 String 属性值。

```
text_handle = findobj('String','\leftarrowsin(t) = .707');
```

然后重新定义文本的 Position 属性值，将该对象移动到新的位置：

```
set(text_handle,'Position',[3*pi/4,sin(3*pi/4),0])
```

findobj 函数也允许用户通过指定多起始点来限制搜索，当继承表中有许多对象时，这样做将会大大提高搜索的速度。在本例中，假设用户知道所需的文本对象位于当前的坐标轴中，可以用以下语句进行快速搜索：

```
text_handle = findobj(gca,'String','\leftarrowsin(t) = .707');
```

12.3.2　使用句柄操作图形对象

1．对象拷贝

用户可以使用 copyobj 函数将一个对象从一个父对象拷贝到另一个父对象中。新对象与旧对象惟一的不同就是其 Parent 属性和句柄。可以同时将多个对象拷贝到一个新的父对象中，也可以将一个对象拷贝到多个父对象中而不改变当前的父子关系。当用户拷贝一个有子对象的对象时，MATLAB 7.0 同时也拷贝其所有的子对象。

假设用户正在绘制多个数据并且希望标注每个图形点（5π/4,sin(5π/4)）。text 函数将使用字符串{5π/4,sin(5π/4)}和一个右箭头来标注数据点，并将'HorizontalAlignment'属性修改为 right 使得数据点位于文本字符串的右边。

```
text('String','\{5\pi\div4, sin(5\pi\div4)\}\rightarrow',...
        'Position',[5*pi/4,sin(5*pi/4),0],...
        'HorizontalAlignment','right')
```

得到的结果如图 12-10 所示。

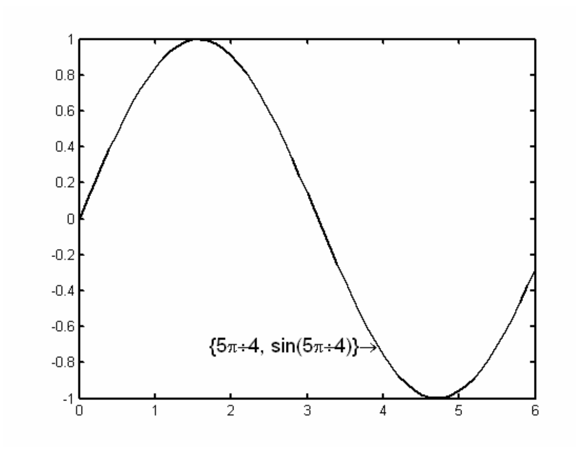


图 12-10　待拷贝的图形

为了使用同样的字符串对另外一个点进行标注，首先使用 copyobj 函数拷贝文本对象。由于以上的语句并没有保存文本对象的句柄，所以用户必须通过 findobj 函数和 String 属性获得句柄，代码设置如下：

```
text_handle = findobj('String',...
                '\{5\pi\div4,sin(5\pi\div4)\}\rightarrow');
```

在创建第二个图形后，通过第一个图形中拷贝标签来为第二个图形添加标签：

```
copyobj(text_handle,gca)
```

2．对象删除

用户可以使用 delete 命令和句柄参数删除一个图形对象。例如，用户可以使用以下语句删除当前的坐标轴（以及其所有子对象）：

```
delete(gca)
```

也可以通过 findobj 命令来获得需要删除的特定对象的句柄。

例如，假设希望删除图 12-10 中的点线，首先获得点线的句柄：

```
line_handle = findobj('LineStyle',':');
```

然后使用这个句柄删除由上述语句获得的线：

```
delete(line_handle)
```

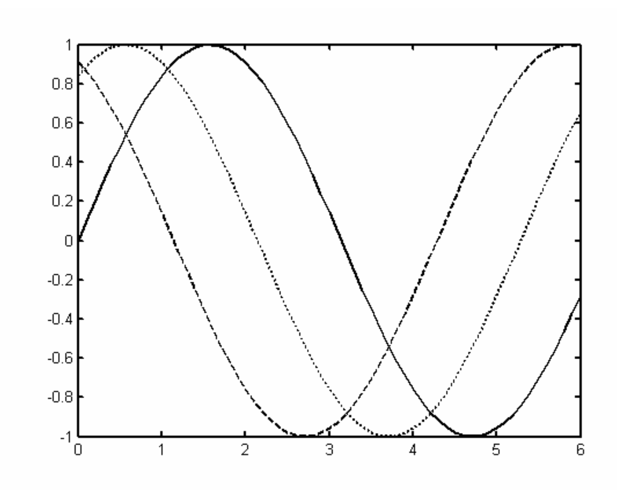


图 12-11　待删除的点线图形

如果将以上两个语句并为一个，代码设置如下：

```
delete(findobj('LineStyle',':'))
```

删除结果如图 12-12 所示。

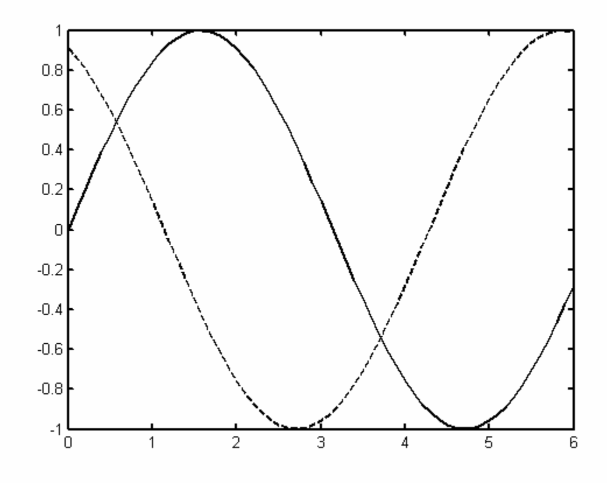


图 12-12　利用句柄删除点线后的图形

12.3.3 控制图形输出

MATLAB 7.0 允许在同一次运行过程中打开多个图形窗口，所以当一个 MATLAB 7.0 程序将创建图形窗口来显示图形用户界面并绘制数据时，有必要对某些图形窗口进行保护，以免成为图形输出的目标，而相应的输出窗口要做好接受新图形的准备。以下内容将讨论如何使用句柄来控制 MATLAB 7.0 显示图形的目标和方法。

1．指定输出目标

默认情况下，MATLAB 7.0 图形创建函数在当前的图形窗口和坐标轴中显示图形。用户可以通过在图形创建函数中使用明确的 Parent 属性直接指定图形的输出位置。例如：

```
plot(1:10,'Parent',axes_handle)
```

在上述代码中，axes_handle 是目的坐标轴的句柄。

uicontrol 和 uimenu 函数指定 Parents 属性的语法比较简单：

```
uicontrol(Figure_handle,...)
uimenu(parent_menu_handle,...)
```

默认情况下，产生图形输出的函数将在当前的图形窗口中显示该图形而不擦除或重置当前窗口的属性。然而，如果图形对象是坐标轴的子对象，为了显示这些图形将会擦除坐标轴并重置坐标轴的大多数属性。用户可以通过设置图形窗口和坐标轴的 NextPlot 属性来改变 MATLAB 7.0 的这种行为。MATLAB 7.0 高级图形函数在绘制图形前先要检查 NextPlot 属性，然后来决定是添加还是擦除重置图形和坐标轴。而低级对象创建函数则不检查 NextPlot 属性，只是简单地在当前窗口和坐标轴中添加新地图形对象。

在表 12-2 中列出了 NextPlot 属性的可能取值。

表 12-2　　　　NextPlot 属性地可能取值

NextPlot	**图形窗口**	**坐标轴**
Add	添加新的图形而不擦除或重置当前窗口	添加新的图形而不擦除或重置当前坐标轴
Replacechildren	删除所有子对象但不重置窗口属性，等同于 clf 函数	删除所有子对象但不重置坐标轴属性，等同于 cla 函数
Replace	删除所有子对象并将窗口重置为默认属性，等同于 clf 函数	删除所有子对象并将坐标轴重置为默认属性，等同于 cla 函数

hold 命令提供访问 NextPlot 属性的简便方法，以下语句将图形和坐标轴的 NextPlot 属性都设置为 add：

```
hold on
```

以下语句将图形和坐标轴的 NextPlot 属性都设置为 replace：

```
hold off
```

MATLAB 7.0 提供 newplot 函数来简化图形 M 文件设置 NextPlot 属性的编写过程。newplot 函数首先检查 NextPlot 属性值，然后根据属性值决定相应的行为。用户应该在所有调用图形创建函数的 M 文件的开头定义 newplot 函数。当用户调用 newplot 函数时，有可能发生以下行为。

（1）检查当前图形窗口的 NextPlot 属性

- 如果不存在图形窗口，则创建一个窗口并设该窗口为当前窗口；

- 如果 NextPlot 值为 add，将该窗口设置为当前窗口；
- 如果 NextPlot 值为 replacechildren，删除窗口的子对象并设置该窗口为当前窗口；
- 如果 NextPlot 值为 replace，删除窗口的子对象，重置窗口属性为默认值，并设置该窗口为当前窗口。

（2）检查当前坐标轴的 NextPlot 属性

- 如果不存在坐标轴，则创建一个坐标轴并设该坐标轴为当前坐标轴；
- 如果 NextPlot 值为 add，将该坐标轴设置为当前坐标轴；
- 如果 NextPlot 值为 replacechildren，删除坐标轴的子对象，并设置该坐标轴为当前坐标轴；
- 如果 NextPlot 值为 replace，删除坐标轴的子对象，重置坐标轴属性为默认值，并设置该坐标轴为当前坐标轴。

默认情况下，图形窗口的 NextPlot 值为 add，坐标轴的 NextPlot 值为 replace。

为了说明 newplot 的使用方法，下面给出一个类似与 plot 的绘图函数 my_plot，该函数在绘制多个图形时将循环使用不同的线型，而不是使用不同的颜色，具体代码设置如下：

```
function my_plot(x,y)
% newplot 返回当前坐标轴的句柄
cax = newplot;
LSO = ['- ';'--';': ';'-.'];
set(cax,'FontName','Times','FontAngle','italic')
set(get(cax,'Parent'),'MenuBar','none')
line_handles = line(x,y,'Color','b');
style = 1;
for i = 1:length(line_handles)
    if style > length(LSO), style = 1;end
    set(line_handles(i),'LineStyle',LSO(style,:))
    style = style + 1;
end
grid on
```

函数 my_plot 使用低级函数 line 语法来绘制数据，虽然 line 函数并不检查图形窗口和坐标轴的 NextPlot 属性值，但是 newplot 的调用使得函数 my_plot 与高级函数 plot 执行相同的操作，即每一次用户调用该函数时，函数都对坐标轴进行清除和重置。my_plot 函数是使用 newplot 函数返回的句柄来访问图形窗口和坐标轴。该函数还设置了坐标轴的字体属性并禁止使用图形窗口的菜单。调用 my_plot 函数的绘图结果如图 12-13 所示。

```
my_plot(1:10,peaks(10))
```

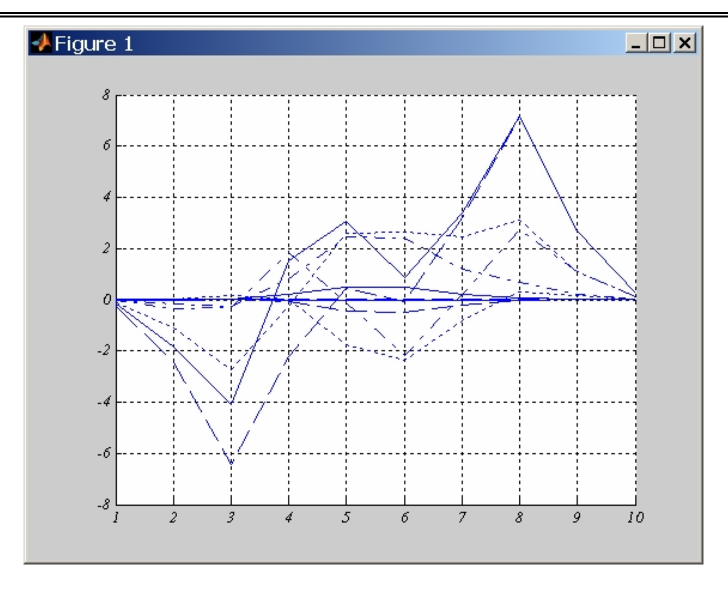


图 12-13　my_plot 函数的典型输出结果

2．保护图形和坐标轴

有些情况下需要对图形窗口和坐标轴进行保护以免其成为图形输出的目标。用户可以将特定窗口或坐标轴的句柄从句柄列表中删除，使 newplot 和其他返回或参考句柄的函数（gca，gco，cla，clf，close 和 findobj）无法找到该句柄，从而保证该窗口不会成为其他程序的输出目标。HandleVisibility 和 ShowHiddenHandles 两个属性可以用来控制保护对象句柄的可见与否。

HandleVisibility 是一个所有对象都具备的属性，该属性值可以为以下取值：

- on：对象句柄可以被任意函数获得，这是该属性的默认值；
- callback：对象句柄对所有在命令行中执行的函数隐藏，而对所有回调函数（用户行为处理函数）总是可见的。这种可见程度将保证用户在命令行中键入的命令不会影响被保护对象；
- off：句柄对所有函数，无论是命令行中执行的还是回调的函数，都是隐藏的。

例如，如果一个用户图形界面以文本字符串形式接受用户的输入，并在回调函数中对这个字符串进行处理，那么如果不对窗口进行保护，字符串“close all”有可能会导致用户图形界面的销毁。为了防止这种情况，用户可以将该窗口关键对象的 HandleVisibility 数值暂时设置为 off：

```
user_input = get(editbox_handle,'String');
set(gui_handles,'HandleVisibility','off')
eval(user_input)
set(gui_handles,'HandleVisibility','commandline')
```

如果一个被保护的图形窗口是屏幕最顶层的窗口，而且在它之下存在未保护的窗口，那么使用 gcf 将返回最高层的未被保护的窗口，gca 的情况与之相同。如果不存在未保护的窗口或坐标轴，MATLAB 7.0 将创建一个窗口并返回它的句柄。

root 对象的 ShowHiddenHandles 属性是句柄可视的使能控制属性。ShowHiddenHandles

的默认值为 off。当该属性值为 on 时，句柄对所有函数都是可视的。当用户希望访问某一特定时刻存在的所有对象时，该属性将十分有用。这里需要说明的一点是，close 函数可以通过使用 hidden 选项来访问不可见的窗口，例如下面的示例代码：

```
close('hidden')
```

即使该窗口是被保护的，该语句也将关闭屏幕最顶端的窗口。使用以下语句将关闭所有窗口：

```
close('all','hidden')
```

有些情况下需要改变坐标轴的可视外观来适应新的图形对象。为了与 MATLAB 7.0 的高级函数保持一致，在改变坐标轴和图形窗口之前，最好先测试一下 hold 属性是否为 on，当 hold 属性为 on 时，坐标轴和图形窗口的 NextPlot 数值均为 add。以下的 my_plot3 函数将接收三维数据并使用 ishold 来检查 hole 属性的状态，以此来决定是否更改视图。

```
function my_plot3(x,y,z)
cax = newplot;
%检测当前的 hold 状态
hold_state = ishold;
LSO = ['- ';'--';': ';'-.'];
if nargin == 2
    hlines = line(x,y,'Color','k');
%如果 hold 为 off 时，改变视图
    if ~hold_state
        view(2)
    end
elseif nargin == 3
    hlines = line(x,y,z,'Color','k');
%如果 hold 为 off 时，改变视图
    if ~hold_state
        view(3)
    end
end
ls = 1;
for hindex = 1:length(hlines)
    if ls > length(LSO),ls = 1;end
    set(hlines(hindex),'LineStyle',LSO(ls,:))
    ls = ls + 1;
end
```

如果 hold 属性为 on，调用 my_plot3 时将不改变视图，否则有 3 个输入参数， MATLAB 7.0 将视图由二维变为三维。

3．关闭请求

定义 CloseRequestFcn 属性使用户能够组织或延迟图形窗口的关闭以及 MATLAB 7.0 运

行的终止。发生以下几种情况时，MATLAB 7.0 将执行由图形窗口的 CloseRequestFcn 属性定义的回调函数（称为关闭请求函数）：

- 在图形窗口中调用 close 命令;
- 用户退出 MATLAB 7.0 时还存在可见的图形窗口（如果一个窗口的 Visible 属性值为 off，则退出 MATLAB 7.0 时并不执行关闭请求函数，而是删除该图形窗口）;
- 使用窗口系统的关闭菜单或按钮来关闭图形窗口。

默认的关闭请求函数保存在一个名为 closereq.m 的 M 文件中，该函数包括以下语句：

```
shh=get(0,'ShowHiddenHandles');
set(0,'ShowHiddenHandles','on');
delete(get(0,'CurrentFigure'));
set(0,'ShowHiddenHandles',shh);
```

这个关闭请求函数通过设置 root 对象的 ShowHiddenHandles 属性使得 HandleVisibility 属性的设置无效，即将所有图形窗口的句柄变成可见。这样，当用户退出 MATLAB 7.0 时，当前图形窗口的关闭请求函数将被调用，在关闭请求函数中删除该图形窗口，然后 root 对象 children 属性列表中的下一个窗口将变成当前窗口，执行它的关闭请求函数，如此重复该过程直至所有的窗口都被删除。此时，如果 MATLAB 7.0 不删除图形窗口， MATLAB 7.0 不能退出。

如果在视图关闭窗口时关闭请求函数发生错误，那么 MATLAB 7.0 将终止退出操作。但是如果使用 quit 或 exit 命令退出 MATLAB 7.0，即使关闭请求函数发生错误，MATLAB 7.0 也会无条件地关闭窗口。delete 函数也总是无视 CloseRequestFcn 属性的取值而无条件地关闭指定的窗口，例如下面的示例代码：

```
delete(get(0,'Children'))
```

上面的语句将删除所有句柄可见的窗口（即所有 HandleVisibility 属性为 on 的窗口）。如果用户希望删除所有窗口（不论句柄是否隐藏），那么可以通过设置 root 对象的 ShowHiddenHandles 属性为 on 来实现。例如下面的语句：

```
set(0,'ShowHiddenHandles','yes')
delete(get(0,'Children'))
```

将无条件地删除所有图形窗口。

12.3.4　在 M 文件中保存句柄

所有句柄无论可见与否都能够保持其有效性，如果用户确实知道某对象的句柄，无论该句柄可见与否，用户都可以使用句柄进行对象属性的设置或查询。

图形 M 文件经常使用句柄来访问属性值，并通过句柄直接定义图形输出的目标。MATLAB 7.0 提供一些有用的函数来获得图形关键对象（例如当前窗口和坐标轴）的句柄，然而在 M 文件中，使用这些函数并不是获得句柄最好的方法。这是因为，在 MATLAB 7.0 中查询对象句柄或其他信息的执行效率并不高，最好还是将句柄直接保存在一个变量中进行引用。另外，由于当前的坐标轴、图形窗口或对象有可能因为用户的交互而发生变化，查询方式难以确保句柄完全正确，而使用句柄变量则可以保证正确地反映对象发生的变化。

为了保存句柄信息，通常用户在 M 文件的开始处保存 MATLAB 7.0 相关的状态信息。

例如，用户可以使用以下语句作为 M 文件的开头：

```
cax = newplot;
cfig = get(cax,'Parent');
hold_state = ishold;
```

这样就无需在每次需要这些信息时都重新进行查询。如果用户在 M 文件中暂时改变了保持状态，用户应当将 NextPlot 的当前属性值保存下来，以便以后重新设置。

```
ax_nextplot = lower(get(cax,'NextPlot'));
fig_nextplot = lower(get(cfig,'NextPlot'));
set(cax,'NextPlot',ax_nextplot)
set(cfig,'NextPlot',fig_nextplot)
```

有一些内置的函数可以修改坐标轴的属性以实现某种特定的效果，这些函数可能会对用户的 M 文件产生一定的影响。

表 12-3 列出了一些 MATLAB 7.0 的内置函数以及它们所修改的属性。注意这些属性仅在 hold 设置为 off 时才会发生变化。

表 12-3　　某些 MATLAB 7.0 的内置函数修改的属性

函数名	坐标轴属性（改变后）	函数名	坐标轴属性（改变后）
fill	Box: on CameraPosition: 2-D view CameraTarget: 2-D view CameraUpVector: 2-D view CameraViewAngle: 2-D view	plot	Box: on CameraPosition: 2-D view CameraTarget: 2-D view CameraUpVector: 2-D view CameraViewAngle: 2-D view
Fill3	CameraPosition: 3-D view CameraTarget: 3-D view CameraUpVector: 3-D view CameraViewAngle: 3-D view XScale: linear YScale: linear ZScale: linear	plot3	CameraPosition: 3-D view CameraTarget: 3-D view CameraUpVector: 3-D view CameraViewAngle: 3-D view XScale: linear YScale: linear ZScale: linear
image (high-level)	Box: on Layer: top CameraPosition: 2-D view CameraTarget: 2-D view CameraUpVector: 2-D view CameraViewAngle: 2-D view XDir: normal XLim: [0 size(CData,1)]+0.5 XLimMode: manual YDir: reverse YLim: [0 size(CData,2)]+0.5 YLimMode: manual	semilogx	Box: on CameraPosition: 2-D view CameraTarget: 2-D view CameraUpVector: 2-D view CameraViewAngle: 2-D view XScale: log YScale: linear
loglog	Box: on CameraPosition: 2-D view CameraTarget: 2-D view CameraUpVector: 2-D view CameraViewAngle: 2-D view XScale: log YScale: log	Semilogy	Box: on CameraPosition: 2-D view CameraTarget: 2-D view CameraUpVector: 2-D view CameraViewAngle: 2-D view XScale: linear YScale: log

12.4 GUI 设计向导

一个可发布的应用程序通常都需要具备一个友好的图形界面，本章就来介绍创建图形用户界面（GUI）的具体方法。本章首先对 GUI 的基本概念作简单的介绍，然后说明 GUI 开发环境 GUIDE 及其组成部分的用途和使用方法，最后说明创建一个完整 GUI 的详细步骤。

12.4.1 图形用户界面概述

通常在开发一个实际的应用程序时都会尽量做到界面友好，最为常用的方法就是使用图形界面。提供图形用户界面的应用程序能够使用户的学习和使用更为方便容易。用户不需要知道应用程序究竟怎样执行各种命令的，而只需要了解可见界面组件的使用方法；用户也不需要知道命令是怎样执行的，只要通过与界面交互就可以使指定的行为得以正确执行。

在 MATLAB 7.0 中，图形用户界面是一种包含多种对象的图形窗口。用户必须对每一个对象进行界面布局和编程，从而使用户在激活 GUI 每个对象时都能够执行相应的行为。另外，用户必须保存和发布所创建的 GUI，使得 GUI 能够真正地得到应用。

MATLAB 为用户开发图形界面提供了一个方便高效的集成开发环境，MATLAB 图形用户界面开发环境 GUIDE（MATLAB Graphical User Interface Development Environment）。上述所有工作都能够使用 GUIDE 方便地实现。GUIDE 主要是一个界面设计工具集，MATLAB 7.0 将所有 GUI 支持的用户控件都集成在这个环境中并提供界面外观、属性和行为响应方式的设置方法。GUIDE 将用户保存设计好的 GUI 界面保存在一个 FIG 资源文件中，同时还能够生成包含 GUI 初始化和组件界面布局控制代码的 M 文件。这个 M 文件为实现回调函数（当用户激活 GUI 某一组件时执行的函数）提供了一个参考框架。虽然使用用户自己编写的包含 GUI 所有发布命令的 M 文件也能够实现一个 GUI，但是使用 GUIDE 执行效率更高。使用 GUIDE 不但能够交互式地进行组件界面布局，而且能够生成两个用来保存和发布 GUI 的文件。

- FIG 文件：该文件包括 GUI 图形窗口及其所有后裔的完全描述，包括所有对象的属性值。FIG 文件式一个二进制文件，调用 hgsave 命令或界面设计编辑器的 File 菜单 Save 选项保存图形窗口时将生成该文件。FIG 文件包含序列化的图形窗口对象，在用户打开 GUI 时，MATLAB 7.0 能够通过读取 FIG 文件重新构造图形窗口及其所有后裔。所有对象的属性都被设置为图形窗口创建时保存的属性。默认情况下，即使用户使用 hgsave 和 hgload 命令保存系统默认的图形工具条和菜单，FIG 文件也不保存这些默认信息。FIG 文件的作用之一就是对象句柄的保存和引用。可以使用 open、openfig 和 hgload 命令类来打开一个后缀为.fig 的文件；
- M 文件：该文件包括 GUI 设计、控制函数以及定义为子函数的用户控件回调函数，主要用于控制 GUI 展开时的各种特征。该 M 文件可分为 GUI 初始化和回调函数两个部分，用户控件的回调函数根据用户与 GUI 的具体交互行为分别调用。这里将 GUI 的 M 文件称为应用程序 M 文件。应用程序 M 文件使用 openfig 命令来显示 GUI。注意应用程序 M 文件并不包括用户界面设计的任何代码，这些代码将完全由 FIG 文件保存。

GUIDE 可以根据用户 GUI 的版面设计过程直接自动生成 M 文件框架，这样就简化了 GUI 应用程序的创建工作，用户可以直接使用这个框架来编写自己的函数代码。这样的编写

方法具有以下优点：

- 应用程序 M 文件已经包含实现一些有用的函数代码，无需用户自行编写；
- 可以使用该 M 文件生成的有效方法来管理图形对象句柄并执行回调函数子程序；
- 提供管理全局数据的途径；
- 文件支持自动插入回调函数原型，确保当前 GUI 与未来发布版本的兼容性。

用户也可以选择由 GUIDE 生成 FIG 文件、自己编写应用程序 M 文件的 GUI 创建方式。编写 M 文件时要注意，应用程序 M 文件中不能包含用户控件的创建命令，所有的界面设计信息都保存在由界面设计编辑器生成的 FIG 文件中。

实现一个 GUI 主要包括 GUI 界面设计和 GUI 组件编程两项工作。整个 GUI 的实现过程可以分为以下几步：

- 通过设置 GUIDE 应用程序的选项来进行 GUIDE 组态；
- 使用界面设计编辑器进行 GUI 界面设计；
- 理解应用程序 M 文件中所使用的编程技术；
- 编写用户 GUI 组件行为响应控制（即回调函数）代码。

12.4.2　启动 GUIDE

在 MATLAB 7.0 中，GUIDE 提供了多种设计模板以方便用户轻松地定制属于自己的 GUI。这些模板均包括了相关的回调函数，用户可以打开它所对应的 M 文件，看到它们的工作方式，或修改相应的函数，从而实现自己需要的功能。

在 MATLAB 7.0 中，可以通过如下两种方法来访问模板：

- 直接输入 GUIDE 命令，打开如图 12-14 所示的界面；
- 如果 GUIDE 已经打开，通过 File 菜单中的 New 选项也可以打开 GUI 模板的设置界面。

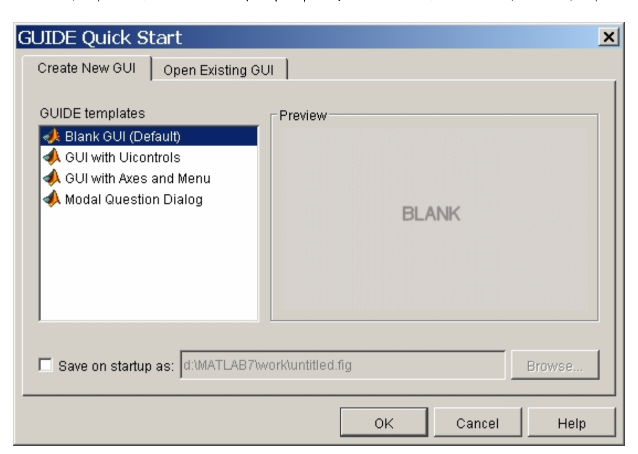


图 12-14　GUI 模板设置界面

在模板设计界面中，用户可以选择创建新的 GUI 或者打开原有的 GUI。在创建新的 GUI 中，MATLAB 7.0 提供了空白模板、带有控制按钮的模板、带有坐标轴和菜单的模板以及问答式对话框 4 种模板。其中空白模板如图 12-15 所示。

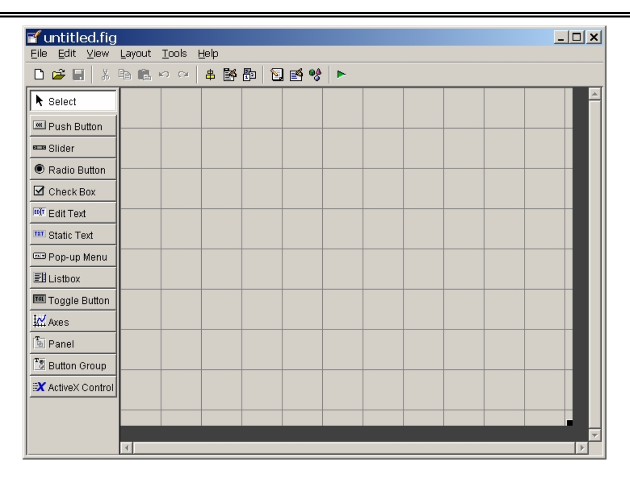


图 12-15 空白界面设计编辑器外观

12.4.3 GUIDE 提供的用户控件

在空白模板中 GUIDE 提供了用户界面控件以及界面设计工具集来实现用户界面的创建工作。用户界面控件分布在界面设计编辑器的左侧，如图 12-16 所示。



图 12-16 各种控件图示

下面简单介绍各种控件的概念和特点。

- 按钮：通过鼠标单击按钮可以实现某种行为（按钮下陷和弹起等）并调用相应的回调子函数；
- 滚动条：使用户能够通过移动滚动条来改变指定范围内的数值输入，滚动条的位置代表用户输入的数值；
- 单选按钮：单选按钮与按钮的执行方式没有本质上的区别，但是单选按钮通常以组为

单位，一组单选按钮之间是一种互相排斥的关系，也就是说任何时候一组单选按钮中只能有一个有效；
- 复选框：复选框与单选按钮类似，只是多个复选框可以同时有效。复选框为用户提供一些可以独立选择的选项进行设置程序模式，例如显示工具条与否以及生成回调函数原型与否等；
- 编辑框：编辑框是控制用户编辑或修改字符串的文本域，其 String 属性包含用户输入的文本信息。如果一个编辑框有输入焦点，在 UNIX 系统中，单击图形窗口的菜单栏，编辑框回调函数就会被调用，而在 Windows 系统下，单击图形窗口不会导致编辑框回调函数的执行。这个区别是处于操作平台方便性考虑而定的；
- 静态文本：静态文本通常作为其他控件的标签使用，用户不能采用交互方式修改静态文本或调用响应的回调函数；
- 弹出式菜单：弹出式菜单将打开并显示一个由其 String 属性定义的选项列表。当用户希望提供一些相互排斥的选项，但不希望使用一系列占用有限空间的单选按钮时，文本菜单将非常有用；
- 列表框：列表框显示由其 String 属性定义的一系列列表项，并使用户能够选择其中的一项或多项；
- 拴牢按钮：拴牢能够产生一个二进制状态的行动（on 或 off）。单击该按钮将使按钮的外观保持下陷状态，同时调用响应的回调函数。再次单击该按钮将使按钮弹起，同时也要调用回调函数。拴牢按钮的回调函数首先要对按钮的状态进行查询，然后才能决定相应的行为；
- 坐标轴：坐标轴使用户的 GUI 可以显示图片，像所有的图像对象一样，坐标轴可以设置许多关于外观和行为的参数；
- 组合框：组合框是图形窗口中的一个封闭区域，它把相关联的控件（例如一组单选按钮）组合在一起，使得用户界面更容易理解；组合框可以有自己的标题以及各种边框；
- 按钮组：按钮组类似于组合框，但是它可以响应关于单选按钮以及拴牢按钮的高级属性，例如单选按钮之间的互斥属性。

注意：后边两种为 MATLAB 7.0 中新增的控件。

12.4.4　界面设计工具集

GUIDE 提供了一个界面设计工具集来实现图形用户界面的创建工作，这些界面设计工具包括：
- 界面设计编辑器：添加并排列图形窗口中的组件对象；
- 属性检查器：检查并设置组件的属性值；
- 图像浏览器：观察此次 MATLAB 7.0 运行过程中图形对象的句柄集成关系表；
- 菜单编辑器：创建窗口菜单和文本菜单。

1．界面设计编辑器

界面设计编辑器使用户能够从组件面板种选择 GUI 组件并将它们排列在图形窗口中。界

面设计编辑器由组件面板、工具栏、菜单栏和界面区域 4 个部分组成。组件面板包含用户界面可获得的所有组件（用户控件对象）；工具栏和菜单栏可以用来启动其他界面设计工具，例如菜单编辑器；界面区域实际上就是激活后的 GUI 图形窗口。

（1）组件面板

在 GUI 界面中放置组件，首先单击组件面板重的按钮，选择需要放置的组件类型，光标变为十字形后使用十字形光标的中心点来确定组件左上角的位置，或者通过在界面区域内单击并拖动鼠标来确定组件的大小。图 12-17 给出了一个添加组件的示例。

GUI 组件布置工作完成后，用户可以使用激活按钮或选择 Tools 菜单的 ActivateFigure 选项来观察 GUIDE 的设计结果。激活图形窗口将发生以下事件：首先保存 FIG 文件和 M 文件，如果用户尚未保存该设计结果，GUIDE 将打开“保存：对话框使用户选择将要创建的 M 文件名；然后 GUIDE 保存与 M 文件同名的 FIG 文件（扩展名为.fig），如果存在一个同名的 M 文件，GUIDE 将会显示一个提示对话框，如果用户选择提示对话框中的 Replace 按钮，原来的 M 文件将被替换；如果选择：Append”，则 GUIDE 将向原有的 M 文件中插入未保存的新组件回调函数并根据应用程序选项对话框设置的变化来修改原有代码。

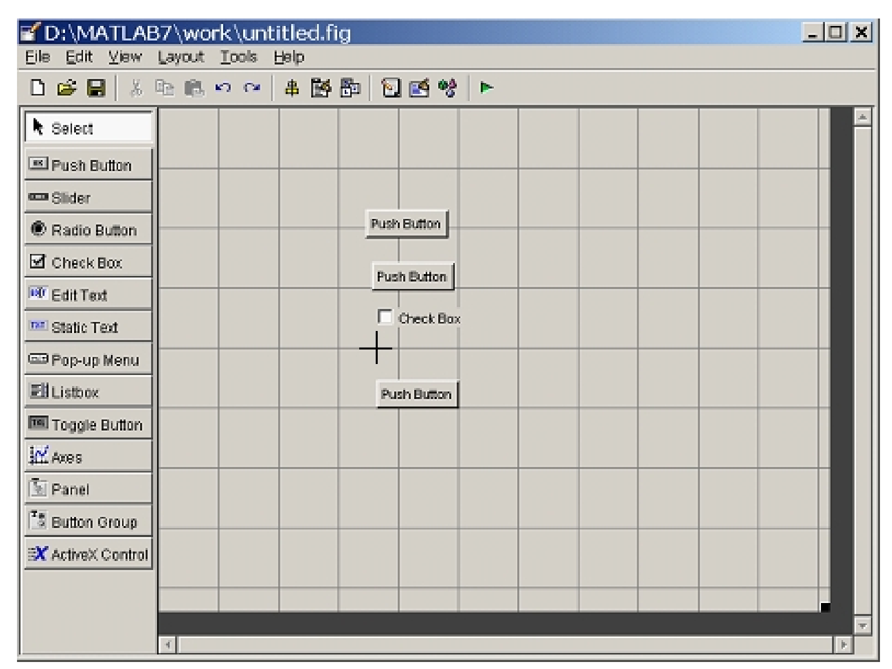


图 12-17　在界面中添加组件的示例

> 注意：如果 FIG 文件和 M 文件已经存在，那么当用户激活 GUI 时，GUIDE 将自动保存应用程序 M 文件和 FIG 文件，因此不要随意激活 GUI 以免有用的组件被替换。

（2）文本菜单

使用界面设计编辑器进行界面设计时，可以先选择一个对象，然后单击鼠标右键来显示与所选对象相连的文本菜单。图 12-18 表示了一个与图形窗口对象联系在一起的文本菜单，所有已定义的回调子函数都列举在该菜单的下方。

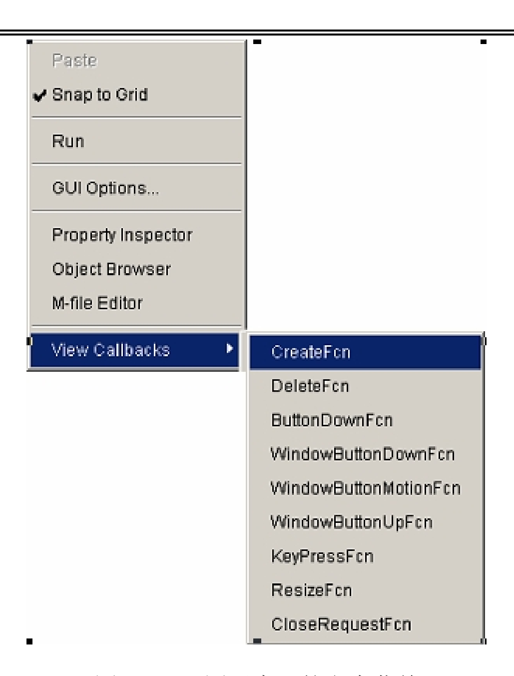


图 12-18　图形窗口的文本菜单

图 12-19 描述了一个与按钮相联系的文本菜单，同样，所有定义的回调函数都列举在该菜单的下方。

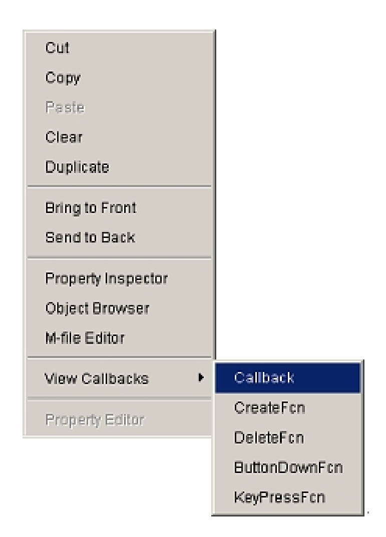


图 12-19　GUI 用户控制文本菜单

（3）排列工具

用户可以在界面区域内通过选择并拖动任意组件或组件群进行组件排列。另外，界面设计编辑器还提供一些更为精确的组件排列方法。

- 排列工具：排列并分布组件群；
- 栅格和标尺：指定任意位置的水平和竖直标线；
- 指引线：指定任意位置的水平和竖直标线；
- 拉前推后：控制组件的前后顺序。

排列工具使用户能够根据其他组件的位置调整被选择组件的间隔或放置某个组件。上述的排列方法可以通过排列工具栏获得，当用户按下“Apply”按钮时，指定的排列操作将应用于所有被选择的组件。图 12-20 显示了排列工具栏的外观。

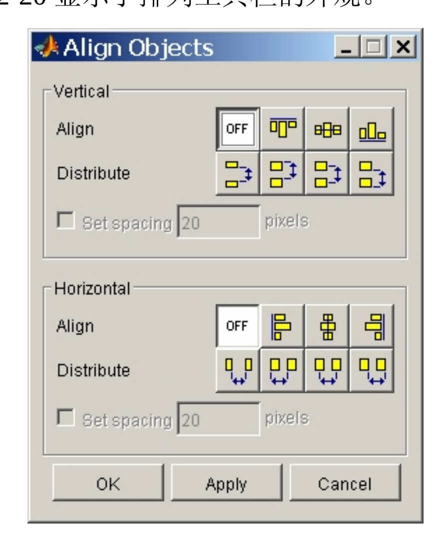


图 12-20 排列工具栏外观

（4）网线和标线

界面区域可以使用网格和标线辅助完成组件设计工作。用户可以将网格线的间隔设置在 10～200 个像素之间，默认情况下以 50 个像素为间隔。如果用户选择了“snap-to-grip”选项，那么对于任何一个在网格线周围 9 像素范围内移动或重画的对象，系统都会自动将该对象放置在该网格线上。无论网格是否可见，snap-to-grip 选项都是有效的。

网格和标线对话框外观如图 12-21 所示。

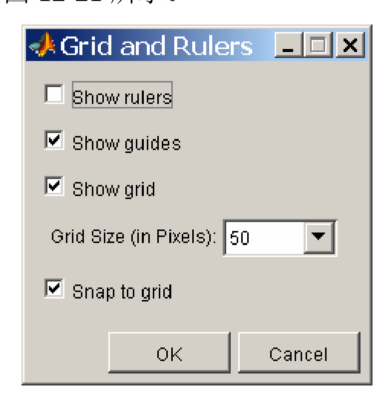


图 12-21 网格和标线对话框外观

该对话框可以完成以下工作。

- 控制标线、网格和指引线的可见与否；
- 设置网格间距；
- 使能 snap-to-grip。

界面设置编辑器有水平和竖直两种指引线。当用户希望在界面设计编辑器的任意位置建立一个组件排列参考标准时，指引线将非常有用。单击标线的左边或顶端并将其拖放到界面

区域中所需位置处就会生成一条指引线。

2．属性检查器

属性检查器提供一个所有可设置属性的列表并显示当前的属性值，使用户能够设置界面中各组件的属性。列表中的每个属性都对应于一个相应于该属性的属性值选择范围。属性检查器的外观如图 12-22 所示。

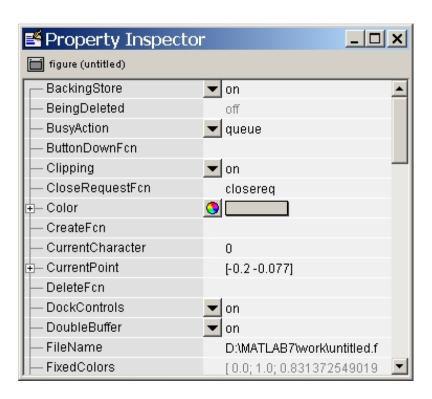


图 12-22　属性检查器外观

用户可以通过多种方式来启动属性检查器。双击界面设计编辑器中的组件；选择 Tools 菜单下的“Property Inspector”选项；选择 Edit 菜单下的“Inspect Property”选项；从组件的文本菜单中选择“Property Inspector”选项。

通过对 GUI 中各个用户控制对象的属性设置可以实现用户所需的控件外观和行为特征。

3．对象浏览器

对象浏览器显示图形窗口中所有对象的继承关系。图 12-23 所为对象浏览器的示例，从图中可以看出，第一个创建的是用户控件组合框，然后是 4 个单选按钮，最后是坐标轴。

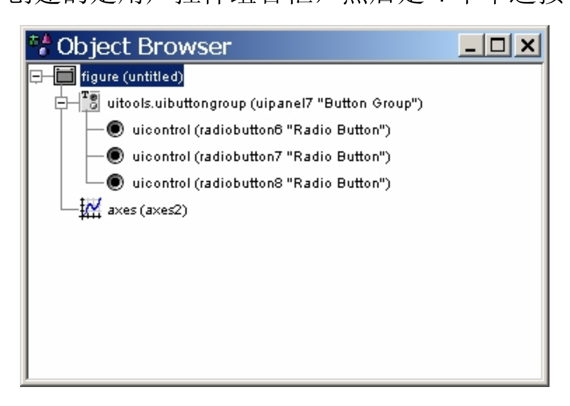


图 12-23　对象浏览器示例

4．菜单编辑器

菜单编辑器的外观如图 12-24 所示。

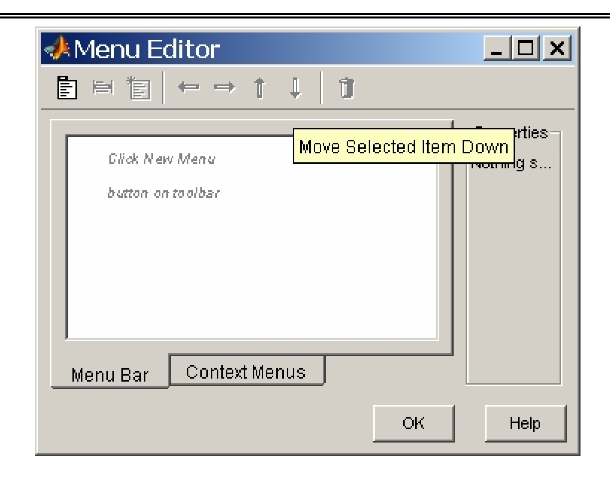


图 12-24　菜单编辑器外观

GUIDE 能够创建两种类型的菜单。在图形窗口菜单栏中显示的菜单栏菜单；当用户在图形对象上单击鼠标右键时弹出的文本菜单。可以使用菜单编辑器来创建这两种类型的菜单。

（1）定义菜单栏菜单

创建菜单的第一步是使用 New Menu 工具栏创建一个菜单，然后来指定菜单的属性。用户单击创建的菜单项将会显示如图 12-25 所示的一个文本域，在该文本域中可以设置菜单的标签、分隔符、选中模式以及回调函数字符串。

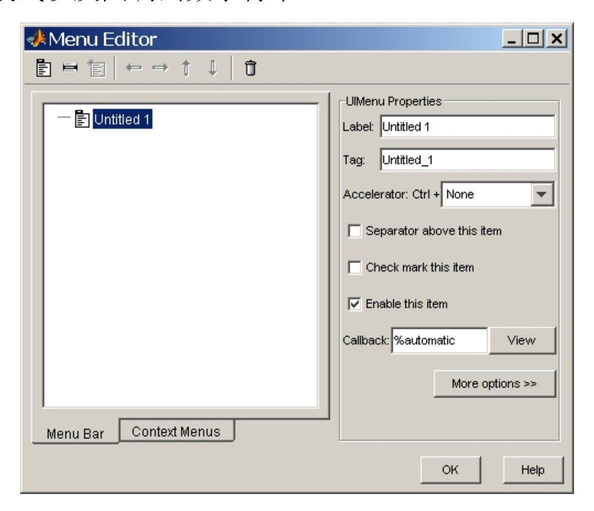


图 12-25　菜单的相关设置

菜单创建成功后，MATLAB 7.0 将该菜单添加到图形窗口的菜单中。

下面将创建菜单项。使用 New Menu Item 工具来添加菜单项，每一个菜单项也可以有级联的子菜单项。图 12-26 显示了为窗口菜单栏定义了 3 个菜单的菜单编辑器外观。

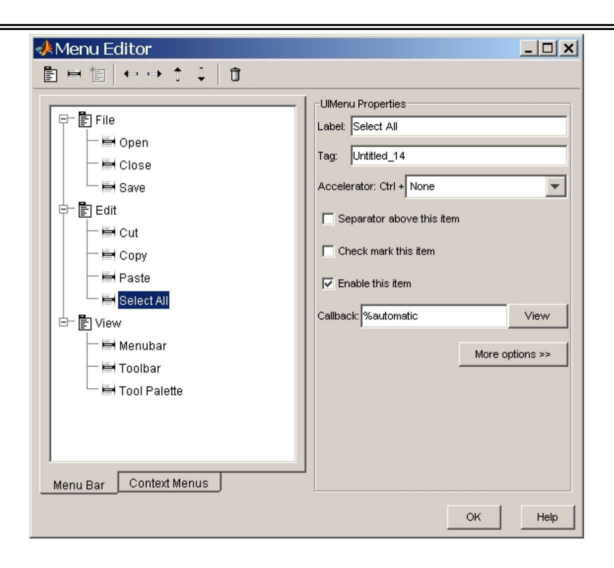


图 12-26　菜单项创建示例

如果用户激活图形窗口，这 3 个菜单将会出现在窗口中，如图 12-27 所示。

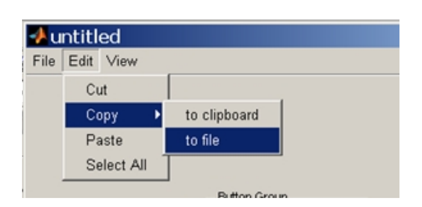


图 12-27　菜单创建结果

（2）定义文本菜单

定义了文本菜单的对象后，当用户单击鼠标右键时，文本菜单随之出现。菜单编辑器能够定义文本菜单并将菜单与对象联系起来。

文本菜单的所有菜单项都是文本菜单的子对象，这些菜单项并不显示在窗口菜单栏中。选择菜单编辑器工具条中的“New Context Menu”来创建父菜单并为之定义一个标签（名称）。注意在定义文本菜单之前要选择菜单编辑器的 Context Menus 标签界面。使用菜单编辑器工具条中的“New Menu Item”按钮来创建文本菜单项，然后给该菜单项添加一个标签并定义回调字符串。

在界面编辑器中选择需要定义文本菜单的对象，使用属性检查器将该对象的 UIContextMenu 属性设置为所需文本菜单的标签名。在应用程序 M 文件中给每个文本菜单项添加一个回调子函数，当用户选择特定的文本菜单项时，这个回调子函数将被调用。

12.4.5　GUI 组态

调用 GUIDE 命令来显示一个空的界面设计编辑器和一个无标题的图形窗口。在为该图形窗口添加组件时，首先应该使用 GUIDE 应用程序选项对话框来进行 GUI 组态。选择界面设计编辑器的 Tools 菜单下的“Application Options”选项打开选项对话框。GUIDE 应用程序选项对话框的外观如图 12-28 所示。

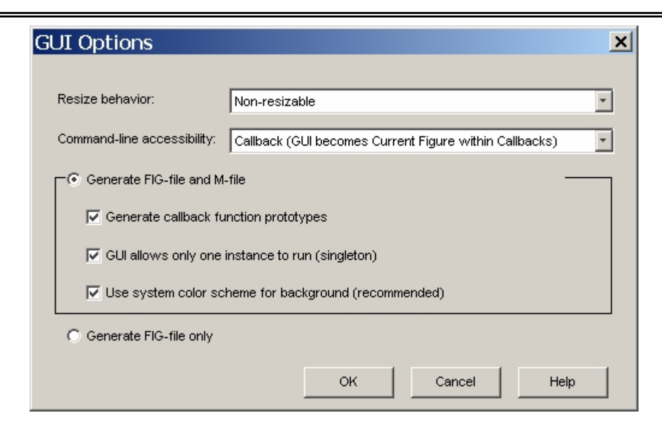


图 12-28　GUIDE 应用程序选项对话框

该对话框能够设置的选项包括以下几项：

- 窗口重画行为（Resize behavior）;
- 命令行访问（Command-Line Accessibility）;
- 生成 FIG 文件和 M 文件（Generate FIG-File and M-File）;
- 生成回调函数原型（Generate Callback Function Prototypes）;
- 同一时刻仅允许运行一个应用程序实例（GUI Allows Only One Instance to Run (Singleton)）;
- 使用系统背景颜色设置（Using the System Background Colors）;
- 仅生成 FIG 文件（Generate FIG-File Only）。

1．重画行为（Resize behavior）

通过选项对话框可以决定用户是否可以重画 GUI 所在的图形窗口以及 MATLAB 7.0 将如何管理重画过程。GUIDE 提供以下 3 种选择：

- Non-resizable: 用户不能改变窗口大小（默认选项）;
- Proportional: 允许 MATLAB 7.0 按照新的图形窗口尺寸自动按比例重新绘制 GUI 组件。注意重画过程中将不改变组件标签字体的大小，如果图像窗口过大，那么组件的标签将不可读。对于在设置过程中始终不关闭的简单 GUI 工具和对话框来说，这种方式非常实用;
- User-specified: 通过编程使重画过程中 GUI 按照用户指定的方式变化。该选项需要用户编写一个 ResizeFcn 属性定义的回调函数，该函数根据新的图形窗口尺寸重新计算组件的大小和位置。

2．命令行访问（Command-Line Accessibility）

在 MATLAB 7.0 中创建一幅图形时，该图形的图形窗口和坐标轴将包括在其父对象的子对象列表中，它们的句柄可以使用诸如 findobj、set 和 get 之类的函数来获得。创建 GUI 时也要创建一个图形窗口，由于绘图命令会直接将结果输出到当前的 GUI 的图形窗口中，所以通常情况下用户不需要额外的 GUI 图形窗口访问，就可以直接将图形输出到窗口中。但是，如果用户需要创建一个包含坐标轴等绘图工具的 GUI，那么访问图形窗口则是必要的。这种情

况下 GUI 要支持命令行的访问。

GUIDE 应用程序选项对话框提供 3 种用户访问权限。

- off：禁止命令行对 GUI 图形窗口的访问；在这种方式下 GUI 图形句柄是隐藏的，这就意味着用户不能够使用 findobj 函数来定位 GUI 中的用户控件句柄。应用程序 M 文件将创建一个对象句柄结构体来保存 GUI 中的所有用户控件句柄并将该结构体传递给子函数保证 GUI 中句柄的使用无误；
- on：允许命令行对 GUI 图形窗口进行访问；
- User-specified：GUI 使用用户设置 Handle Visibility 和 IntergerHandle 这两个图形窗口属性值，它们决定句柄是否能被命令行获得。Handle Visibility 属性决定图形窗口的句柄对视图访问当前图形窗口的命令是否可见；该属性设置为 off 将导致图形窗口的句柄从根对象的子对象列表中删除，使该图形窗口不再是当前图形窗口。但是该图形窗口的句柄仍然有效，明确使用该句柄的函数仍然能够工作。IntergerHandle 决定图形窗口的句柄是一个整数还是浮点数。如果该属性是 off，MATLAB 7.0 将使用一个不可重复使用的浮点数来代替整数，这将大大减小用户对图形窗口的误操作几率。

3．**生成 FIG 文件和 M 文件**（Generate FIG-File and M-File）

如果用户希望 GUIDE 同时创建 FIG 文件和应用程序 M 文件，在 GUIDE 应用程序选项对话框中选择“Generate FIG-File and M-File”选项。一旦用户选择了这个选项，用户就可以选择以下任何一项 M 文件的组态项目。

- 生成回调函数原型（Generate Callback Function Prototypes）：当用户在 GUIDE 用程序选项对话框中选择该选项时，GUIDE 将在应用程序 M 文件中为每一个组件添加一个回调函数（注意组合框和静态文本组件不使用自身的 Callback 属性）。用户必须为回调函数编写代码。GUIDE 还为用户添加了一个与弹出式（右键鼠标出现）文本菜单选项对应的回调子函数，用户无论何时选择菜单项都调用该函数；
- 同一时刻仅允许运行一个应用程序实例（GUI Allows Only One Instance to Run (Singleton)）：该选项允许用户选择图形窗口的两种行为。允许 MATLAB 7.0 的一次运行过程中仅有 GUI 一个实例；允许 MATLAB 7.0 显示 GUI 的多个实例。如果用户允许运行一个实例，那么无论何时执行一个发布 GUI 的命令，MATLAB 7.0 都将重新使用已存在的 GUI 图形窗口。如果 GUI 已存在，MATLAB 7.0 将该 GUI 带到前台而不是重新创建一个新的窗口。如果允许 MATLAB 7.0 显示 GUI 的多个实例，则每一个 GUI 调用命令都将创建一个新的窗口。GUIDE 通过在应用程序 M 文件中生成使用 openfig 命令的代码来实现这个特征，而通过 reuse 或 new 字符来指定 GUI 的一个或多个实例；
- 使用系统背景颜色设置（Using the System Background Colors）：GUI 组件使用的颜色与计算机系统有关。选项使图形窗口的背景颜色与用户控件默认背景颜色（与系统有关）相匹配。图 12-29 中左边的图形窗口未选择该选项，而右边的图形窗口选则了该选项，可以看到二者的效果差异。

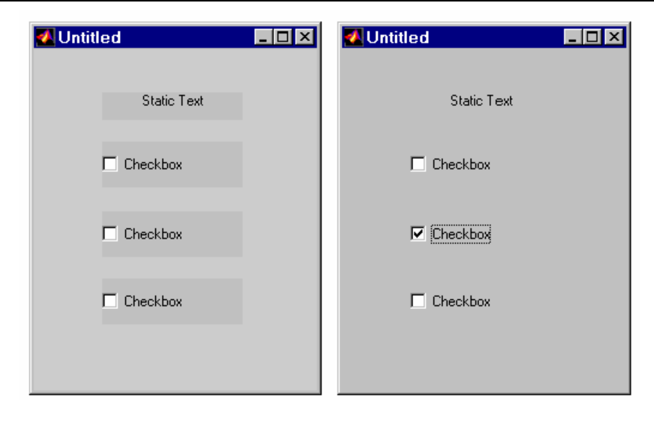


图 12-29　背景颜色使用系统颜色设置与否的比较

如果用户不希望 GUIDE 生成应用程序 M 文件，在 GUIDE 应用程序选项对话框中选择 Generate FIG-File Only 选项。当用户在界面设计编辑器中保存 GUI 时，GUIDE 将仅仅创建能够使用 open 和 hgload 命令重新显示的 FIG 文件。当用户选择这个选项时，必须将每一个 GUI 组件的 Callback 属性设置为一个 MATLAB 7.0 能够理解并执行指定操作的字符串，这个字符串可以是一个表达式也可以是一个 M 文件。如果用户希望生成一个与应用程序 M 文件完全不同的程序范例时可以选择这个选项。

12.4.6　GUI 界面设计

用户可以使用 openfig、open 或 hgload 命令来显示一个 GUI 图形窗口，这些命令将相应的 FIG 文件装载到 MATLAB 7.0 的工作平台中，然后用户就可以对该 GUI 进行重新设计。用户可以使用界面设计编辑器 File 菜单中的“Open”选项或以下命令来装载一个已存在的 GUI 进行编辑：

```
guide mygui.fig
```

GUI 界面设计是通过使用界面设计编辑器进行的。基本的编辑方法在前面已经做了详细介绍，这里就不再赘述。

GUI 的创建过程中经常会要求用户定义组件的 Tag 属性值和 callback 子函数名。GUIDE 要求用户为 GUI 界面中的每个组件指定一个 Tag 属性值，然后根据该字符串来命名回调函数。在第一次激活或保存 GUI 之前最好为组件选择一个 Tag 名和文件名。

使用界面设计编辑器 File 菜单的 Save as 选项可以给应用程序 M 文件重命名并重新设置 callback 属性以保证回调函数的正常运行。注意由于 GUIDE 使用 Tag 属性来命令函数和结构体的域名，所以用户选择的 Tag 必须为一个有效的 MATLAB 7.0 变量名。可以使用 isvarname 函数来确定用户指定的字符串是否有效。如果用户在 GUIDE 中创建 M 文件和 FIG 文件后，对 GUI 进行了修改，那么必须确保用户的代码能够兼容这些修改。

GUIDE 自动给每一个用户控件的 Tag 属性分配一个字符串并使用该字符串构造生成的回调子函数名，并为句柄结构体添加一个域名。

如果用户在 GUIDE 生成回调子函数后对 Tag 属性进行了修改，GUIDE 将不会生成一个新的子函数。然而，由于句柄结构体是实时创建的，所以 GUIDE 将在句柄结构体中使用新的 Tag 名。避免产生问题最好的方法就是在添加组件的同时设置 Tag 属性值。如果一定要在

应用程序 M 文件创建后修改 Tag 并希望重新命名回调子函数来保持 GUIDE 所使用的名称一致，那么用户需要修改句柄结构体所有输出数据参考和修改回调子函数名称。当用户保存或激活 GUI 时，GUIDE 将使用一个在应用程序 M 文件中执行回调子函数的字符串来替换每一个取值为 automatic 的 Callback 属性。当第一次将一个按钮插入界面中时，其 Callback 属性如图 12-30 所示。

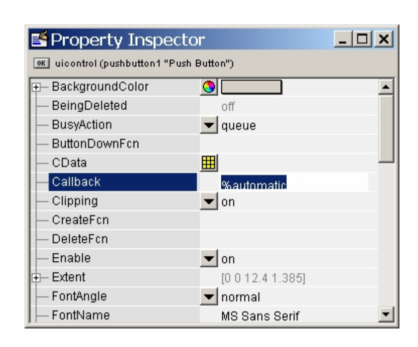


图 12-30　按钮组件的 Callback 属性

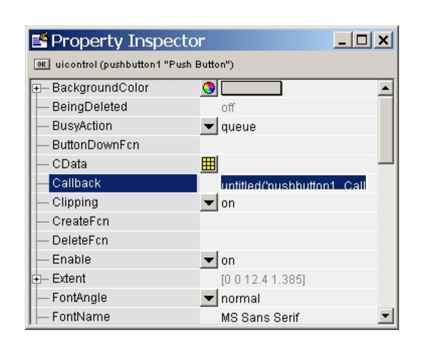


图 12-31　添加第一个按钮的字符串

用户保存或激活图形窗口时，GUIDE 将修改 Callback 属性为执行回调子函数字符串，图 12-31 表明了添加了第一个按钮的字符串。

如果用户希望修改回调子函数的名称，那么必须同时修改用户控件的 Callback 属性字符串。图 12-32 为回调子函数重命名为 Closebutton_Callback 后的外观。

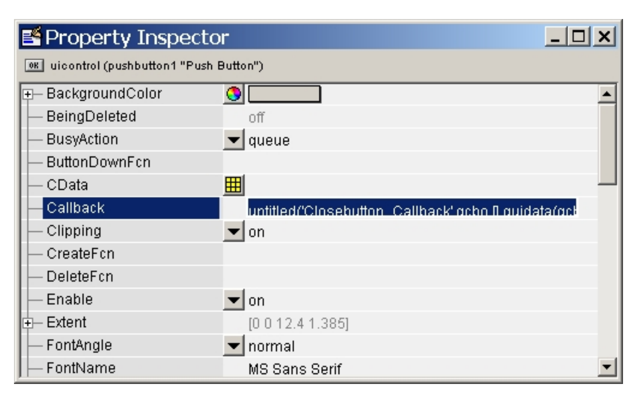


图 12-32　回调子函数重命名后的外观

GUI 的 FIG 文件和相关的应用程序 M 文件的文件名相同，只是文件扩展名不同。当用

户执行 M 文件以发布 GUI 时，使用 mfilename 命令从 M 文件名来确定 FIG 文件名：

```
fig = openfig(mfilename,'reuse');
```

如果 FIG 文件名与 M 文件名不同，以上语句将不会被成功调用。

对于 GUI 实现的第 3 个步骤——回调函数编程将在下一节向读者介绍。

完成 GUI 的创建工作后，使用 File 菜单下的“Save”或“Save as”选项将用户界面保存为一个 FIG 文件。当用户保存或激活图形窗口时，GUIDE 将自动创建应用程序 M 文件。执行 M 文件来显示 GUI。对于那些用户尚未实现，而 GUIDE 在 M 文件中保存原型的回调函数，运行 MATLAB 7.0 时将返回一个消息说明这些函数尚未实现。

12.5 编程设计 GUI

在前面的基础上，本节将主要介绍 GUI 的深入编程方法。首先将说明应用程序 M 文件系统生成代码的具体含义以及利用句柄结构体管理 GUI 数据的方法，然后介绍 GUI 组件回调函数的类型和回调函数的中断方法，最后说明如何控制 GUI 图形窗口的行为。

12.5.1 M 文件以及 GUI 数据管理

1．应用程序 M 文件理解

GUI 包含许多可以使软件与用户中断进行交互的用户界面组件，GUI 的实现任务之一就是控制这些组件如何响应用户的行为。对应用程序 M 文件代码进行详细分析的目的就是要通过了解 GUIDE 创建应用程序 M 文件的功能，从而实现 GUI 的规划。

MATLAB 7.0 通过创建应用程序 M 文件为 GUI 控制程序提供一个框架。这个框架孕育着一种高效而坚固的编程方法，即所有代码（包括回调函数）都包含在应用程序 M 文件中，这就使得 M 文件仅有一个入口可以初始化 GUI 或调用相应的回调函数以及 GUI 中希望使用的任意帮助子程序。无论用户是否使用 GUIDE 来创建应用程序 M 文件，这里所说的编程技术对用户进行 GUI 编程都是有用的。

（1）回调函数自动命名

GUIDE 给添加到应用程序 M 文件中的回调子函数自动命名。GUIDE 还将 Callback 属性值设置为一个字符串使用户激活控件时该子函数能够被调用。

首先说明 GUIDE 如何为回调子函数命名。当用户在 GUI 界面中添加一个组件时，GUIDE 为该组件的 Tag 属性指定一个用来生成回调函数名称的值。例如，假设用户添加到界面中的第一个按钮被称为 pushbutton1，当用户保存或激活图形窗口时，GUIDE 在应用程序 M 文件中添加一个名为 pushbutton1_Callback 的回调子函数。如果用户为该按钮定义了 ButtonDownFcn 属性，则相应的回调子函数的名称为 pushbutton1_ButtonDownFcn。

GUIDE 通过指定回调字符串为回调子函数命名。当用户第一次为 GUI 界面添加一个组件时，其 Callback 属性被设置为字符串<automatic>，当用户保存或激活 GUI 时，该字符串将通知 GUIDE 使应用程序 M 文件中相应的回调子函数名来替换该字符串。

（2）应用程序 M 文件的执行路径

应用程序 M 文件根据 GUI 调用文件时所传递的参数类型来决定有待执行的行为。例如，如果不向 M 文件传递任何参数，则调用 M 文件将会发布该 GUI（如果此时用户指定了一个 M 文件的输出参数，那么 M 文件将返回 GUI 图形窗口的句柄）；如果使用一个子函数名作为传递给 M 文件的第一个参数，那么调用 M 文件将会执行指定的子函数（通常是回调子函数）。

应用程序 M 文件包含一个调度函数，该函数可以使 GUI 能够根据调用方式决定执行路径。应用程序 M 文件调度函数的功能是通过在 if 语句中使用 feval 函数实现的。在调用 M 文件时，feval 函数将执行字符串参数所指定的子函数。feval 函数在一个 try/catch 调试语句块中执行，这是因为当 GUI 视图调用不存在的子函数（找不到与传递参数名称相同的子函数）或调用发生错误时能够得到正确的处理。以下是 GUIDE 生成的调度函数功能代码（用户不能够修改）：

```
%若无输入参数，则打开 GUI
if nargin ==0
    fig = openfig(mfilename,'reuse');
    …
%如果输入参数为字符串，执行相应子程序
elseif ischar(varargin{1})
    try
        [varargout{1:nargout}] = feval(varargin{:});
    catch
    disp(lasterr);
    end
end
```

任何由子函数返回的输出参数都将通过该函数返回 GUI。虽然 GUIDE 生成的回调子函数的参数是明确的，但是参数列表的长度是变化的，这是由于输入参数 varargin 其实可以是多个参数，用户可以在调用 M 文件时通过该参数给被调用的子函数赋予任意多个用户所需的参数。用户可以通过编辑 Callback 属性字符串来传递额外的参数。

（3）GUI 初始化

首先，应用程序 M 文件使用 openfig 命令来装载 GUI 图形窗口，格式如下：

```
fig = openfig(mfilename,'reuse');
```

一定要注意这个语句打开的 FIG 文件名是源于应用程序 M 文件的（mfilename 命令返回当前执行的 M 文件名），如果用户使用由 GUIDE 创建的应用程序 M 文件，那么用户必须保证 FIG 文件与 M 文件同名。参数 reuse 决定任何时候都只能由一个 GUI 实例在运行。

应用程序 M 文件自动包含一些管理 GUI 的有效技术。

- 单个/多个实例控制。当设计 GUI 时，用户必须明确选择是否允许 GUI 图形窗口的多个实例窗口存在。如果选择允许，也就是说同一时刻只能有一个实例存在，那么以后视图创建另一个 GUI 的操作将仅仅导致已存在的 GUI 出现在其他窗口之上。很多信息对话框（尤其是模态对话框）只能同时存在一个实例，因为对于用户的某种特定行为，对话框只能提示一次。GUIDE 界面设计编辑器实际上是一个允许多个实例存在的

GUI，设计这个 GUI 的目的是使用户能够同时打开多个界面；

- GUI 图形窗口在屏幕中的位置不受目标计算机屏幕和分辨率的影响。应用程序 M 文件使用 movegui 命令来确保 GUI 图形在目标计算机的屏幕上可见，使 GUI 不受计算机屏幕的尺寸和分辨率的影响。如果指定的窗口位置将导致 GUI 窗口位于屏幕之外，那么 movegui 命令将窗口移动到屏幕中离指定位置最近的地方。movegui 语句格式如下：movegui(fig,'onscreen')。其中 fig 是由 openfig 命令返回的 GUI 图形窗口句柄；
- 自动创建 GUI 组件句柄结构体；
- 自动命名 Tag 属性、生成子函数原型并指定回调属性字符串。当用户发布 GUI 时，应用程序 M 文件创建一个包含所有 GUI 组件句柄的结构体，然后将该结构体保存在图形窗口的应用程序数据中以备将来的使用。句柄结构体的域名与相应对象的 Tag 属性值一致。

用户可以使用相同的方法访问隐藏的图形窗口句柄，例如假设窗口的 Tag 属性为 figure1，则 handles.figure1 就是图形窗口的句柄。

应用程序 M 文件使用 guihandles 和 guidata 来创建并存储句柄结构体：

```
handles = guihandles(fig);
guidata(fig,handles);
```

注意只有那些 Tag 属性值为有效字符串的对象的句柄才能够保存在句柄结构中。可以使用 isvarname 来确定字符串是否有效。

句柄结构体是传递给所有回调函数的参数之一，因而用户可以使用这个结构体来保存数据并在子函数之间传递。

- 单个 M 文件同时包含 GUI 初始化和回调函数执行代码。

2．GUI 数据管理

GUIDE 使用保存在 GUI 图形窗口中的应用程序来定义和实现数据的存储和获取机制。GUIDE 使用这个机制来存储一个包含所有 GUI 组件对象句柄的结构体。由于这个结构体将传递每一个对象的回调函数，所以该结构体也可以用来保存其他数据。

12.5.2 回调函数的使用方法

1．回调函数类型

实现一个 GUI 的首要机制就是构成用户界面的用户控件的回调函数进行编程。除了用户控件的 Callback 属性，还可以使用其他一些属性来定义回调函数。

（1）所有图形对象的回调函数属性

所有图形对象都有 3 个能够定义回调函数的属性：

- ButtonDownFcn：当用户将鼠标放置在某个对象或对象相邻的 5 个像素范围内时，如果单击鼠标左键，MATLAB 7.0 将会执行回调函数；
- CreatFcn：MATLAB 7.0 将在创建对象时调用回调函数；
- DeleteFcn：MATLAB 7.0 在删除对象之前调用回调函数。

（2）图形窗口的回调属性

图形窗口有如下所述的几种额外的用来执行相应用户行为的属性。

- CloseRequestFcn：当请求关闭图形窗口时 MATLAB 7.0 将执行这个回调函数；
- KeyPressFcn：当用户在图形窗口内按下鼠标键时 MATLAB 7.0 将执行这个回调函数；
- ResizeFcn：当用户重画图形窗口时 MATLAB 7.0 将执行这个回调函数；
- WindowButtonDownFcn：一旦用户在图形窗口内无控件的地方按下鼠标键时，MATLAB 7.0 就会执行这个回调函数；
- WindowButtonMotionFcn：当用户在图形窗口中移动鼠标时，MATLAB 7.0 将执行这个回调函数；
- WindowButtonUpFcn：当用户在图形窗口中释放鼠标键时，MATLAB 7.0 将执行这个回调函数。

MATLAB 7.0 将根据用户的行为来判断究竟执行哪个回调函数。单击一个有效的用户控件将会阻碍任何 ButtonDownFcn 和 WindowButtonDownFcn 回调函数的执行，而如果用户单击一个无效用户控件、图形窗口或其他定义了回调函数的图形对象时，MATLAB 7.0 将首先执行图形窗口的 WindowButtonDownFcn 函数，然后再执行鼠标单击对象的 ButtonDownFcn 函数。

2．回调函数执行中断

默认情况下 MATLAB 7.0 允许正在执行的回调函数被后来调用的回调函数中断。例如，假设用户创建了一个在装载数据时能够显示一个进展条的对话框，这个对话框包含一个取消按钮可以组织数据的装载操作，那么取消按钮的回调函数将会中断正在执行的数据装载子函数。某些情况下用户可能不希望正在执行的回调函数被用户的行为中断，例如，在重新显示一幅图形之前，可能会需要使用一个数据分析工具进行数据流长度计算。假设用户行为可以中断回调函数的执行，此时如果用户无意中单击鼠标使回调函数执行中断，那么就可能导致 MATLAB 7.0 在返回原来的回调函数之前发生状态改变，引起执行错误。

（1）可中断设置

所有图形对象都有一个控制其回调函数能否被中断的属性 Interruptible，该属性的默认值为 on，表示回调函数可以中断。然而 MATLAB 7.0 只有在遇到一些特定的命令（drawnow、figure、getfreame、pause 和 waitfor）时才会执行中断，转而查询事件序列，否则将会继续执行正在执行的回调函数。在回调函数中出现的计算或指定属性值的 MATLAB 7.0 命令将会被立即执行，而影响图形窗口状态的命令将被放置在事件序列中。事件可以由被任何导致图形窗口重画的命令或用户行为引发，例如定义了回调函数的鼠标移动行为。仅仅当回调函数执行完毕或回调函数包含 drawnow、figure、getframe、pause 和 waitfor 命令时，MATLAB 7.0 才进行事件序列的处理。

如果在回调函数的执行过程中遇到上述某个命令，MATLAB 7.0 将先执行的程序挂起，然后处理事件序列中的事件。MATLAB 7.0 控制事件的方式依赖于事件类型和回调函数对 Interruptible 属性的设置。只有在当前回调对象的 Interrupotible 属性值为 on 的情况下，导致其他回调函数执行的事件才可以真正执行回调函数；导致图形窗口重画的事件将无视回调函数的 Interruptible 属性值而无条件地执行重画任务；对象的 DeleteFcn 属性和 CreatFcn 属性或

图形窗口的CloseRequestFcn属性或ResizeFcn属性定义的回调函数将无视对象的Interruptible属性而中断正在执行的回调函数。

所有对象都具有一个BusyAction属性，该属性决定了对于在不允许中断的回调函数执行期间发生的事件的处理方式。BusyAction有两种可能的取值。

- queue: 将事件保存在事件序列中并等待不可中断回调函数执行完毕后处理;
- cancel: 放弃该事件并将事件从序列中删除。

（2）回调函数执行期间的事件处理

以下几种情况描述了MATLAB 7.0在一个回调函数的执行期间是如何处理事件的。

- 如果遇到了drawnow、figure、getframe、pause、waitfor命令中的一个命令，那么MATLAB 7.0将该回调函数挂起并开始处理事件序列;
- 如果事件序列的顶端事件要求重画图形窗口，MATLAB 7.0将执行重画并继续处理事件序列中的下一个事件;
- 如果事件序列的顶端事件将会导致一个回调函数的执行，MATLAB 7.0将判断回调函数被挂起的对象是否可中断。如果回调函数可中断，MATLAB 7.0执行与中断事件相关的回调函数；如果该回调函数包含drawnow、figure、getframe、pause、waitfor命令之一，那么MATLAB 7.0将重复以上步骤；如果回调函数不可中断，MATLAB 7.0将检查事件生成对象的BusyAction属性；如果该属性值为queue，MATLAB 7.0将事件保留在事件序列中；如果cancel则放弃该事件;
- 当所有事件都被处理后，MATLAB 7.0恢复被中断函数的执行。

这些步骤都一直持续到回调函数执行完毕为止。当MATLAB 7.0返回命令窗口时，所有残余的事件都已经被处理了。当然，由于序列中事件的类型不同，以上步骤有可能不一一执行。

12.5.3 图形窗口的行为控制

在设计GUI时，需要考虑显示GUI时图形窗口将怎样展开，以及一个GUI图形窗口的行为是用户所需要的。考虑以下几种情况：

- 一个实现图形注释的工具GUI通常设计成其他MATLAB 7.0执行任务可访问的GUI，这个工具可能一次只能对一幅图形进行注释，所以每一幅图形都需要一个新的工具实例;
- 一个对话框，能向用户发出询问并阻止MATLAB 7.0运行直至用户作出回答。但是用户可能需要通过观察其他的MATLAB 7.0窗口获得信息后才能够对该对话框作出回答。
- 一个警告用户其指定的操作将会破坏文件的对话框，该对话框能够执行用户所需的操作前强迫用户作出回答。此时的图形窗口既被阻止又是模态的（用户不可观察其他窗口）。

以下3种技术能够有效地实现以上GUI的设计要求：

- 允许单个或多个GUI实例同时运行;
- 在显示GUI时阻止MATLAB 7.0的运行;

- 使用模态图形窗口使用户只能与当前执行的 GUI 进行交互。

模态图形窗口能够捕捉在 MATLAB 7.0 窗口任何可见位置处发生的键盘和鼠标事件。这就意味着一个模态图形窗口能够处理用户与任何组件的交互操作，但是不允许用户访问其他的 MATLAB 7.0 窗口（包括命令窗口）。另外，除非删除该模态窗口，否则窗口将始终位于窗口堆栈的最上方。如果用户希望在进行 MATLAB 7.0 其他操作之前必须先响应 GUI，那么最好使用模态图形窗口。

下面说明如何使一个 GUI 窗口模态化。事实上设置图形窗口的 WindowStyle 属性为 modal 就可将图形窗口模态化。用户可以使用属性检查器，也可以通过在应用程序 M 文件的初始化阶段调用以下语句进行属性设置：

```
set(fig,'WindowStyle','modal')
```

在这个语句中，set 函数使用由 openfig 函数返回的句柄实现设置工作。

如果要释放模态窗口的控制权，可以在模态窗口 GUI 的回调函数中使用以下方法之一：

- 删除窗口：delete(figure_handle)
- 使窗口不可见：set(figure_handle,'Visible','off');
- 修改窗口的 WindowStyle 属性值为 normal：set(figure_handle,'WindowStyle','normal')。

用户还可以在一个模态窗口按 ctrl＋C 键将该窗口转换为普通窗口。

无论使用哪一种方法释放模态窗口控制权都需要获得该窗口的句柄。由于大多数的 GUI 将图形窗口句柄隐藏以避免无意识的访问，所以在回调函数中使用 gcbf 命令能够最有效地获取窗口句柄。gcbf 返回当前回调对象所在窗口的句柄，这个句柄就是释放控制权所需的句柄。例如，假设用户对话框包括一个用来关闭对话框的按钮，其回调函数末尾处将调用 delete 来删除对话框，示例代码如下：

```
function varargout = pushbutton1_Callback(h,eventdata,handles,varargin)
%执行程序代码
…
%用户响应对话框后删除窗口
delete(gcbf)
```

12.6　图形用户界面设计实例

实现一个用来显示名字和电话号码的 GUI，该 GUI 中输入的名字和电话号码都保存在一个 MAT 文件中。要求该 GUI 可以添加新条目并将该条目保存在同一个 MAT 文件或一个新的 MAT 文件中。

12.6.1　图形界面的实现

要实现图形界面这个问题要用到以下几种 GUI 编程技术：

- 使用打开和保存对话框的方法为使用者提供寻址和打开地址簿 MAT 文件、保存 MAT 文件修改、创建新地址簿的方法;
- 定义 GUI 菜单回调函数;

- 使用 GUI 句柄结构体保存和传递全局数据（名字和电话号码）;
- 使用 GUI 图形窗口重画函数实现重新显示工作。

这里仅仅介绍 GUI 组态和界面设计工作，关于回调函数的编程方法将在下一节介绍。

下面就介绍其具体解决方法。

步骤一：GUI 组态。打开 GUI 应用程序选项对话框，进行如下设置：

- 重画行为：User-spcified;
- 命令行可访问性：Off;
- 同时生成 FIG 文件和 M 文件;

  生成回调函数原型：Generate callback fuction prototypes;

  同时只允许一个实例运行：Application allows only one instance to run。

步骤二：进行界面设计。在命令行中键入 guide，选择空白模板。

可以按照用户的喜好任意设计 GUI 界面并排列各组件，例如如图 12-33 所示的 GUI 界面。使用“Prev”和“Next”按钮在地址簿中的条目向前或向后翻页。

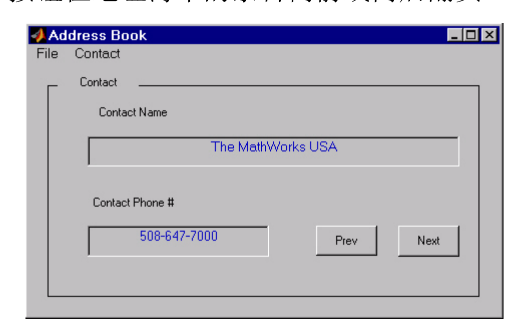


图 12-33　地址簿的 GUI 界面

步骤三：设计菜单。地址簿 GUI 包含一个 File 菜单，其菜单项为 Open，GUI 用该菜单项来装载地址簿 MAT 文件。当用户对地址簿进行修改时，用户需要保存被修改的 MAT 文件或将这些修改保存为一个新的 MAT 文件，使用 File 菜单的“Save”和“Save as”菜单项使用户能够实现这些要求。

步骤四：对各个组件的属性进行设置。在界面设计编辑器中使用菜单命令打开属性检查器，根据用户的界面需要设置组件的 Tag、callback 等属性。必须设置“Prev”和“Next”按钮的 Callback 属性以及 Save 和 Save as 菜单项的回调函数，使它们分别能够调用同一个回调函数 Save_Callback、Prev_Next_Callback。另外，将窗口的 ResizeFcn 属性设置为下面的格式：

```
address_book('ResizeFcn',gcbo,[],guidata(gcbo))
```

步骤五：保存 GUI。激活 GUI 界面，确保界面符合用户要求。设计满意后保存 GUI。

步骤六：执行 GUI。运行地址簿 GUI 的应用程序 M 文件。可以对该 M 文件进行反复调试使之符合用户的要求。

12.6.2　行为控制的实现

前一节已经介绍了地址簿界面的实现，下面来讲如何实现地址簿 GUI 的行为控制。在 GUI 中最重要的技术之一就是保证信息跟踪正确并使不同的子函数能够获得该信息。这里所指的信息包括当前 MAT 文件名，MAT 文件名中的名字和电话号码；一个表明当前名字和电

话号码，一个表明当前姓名和电话号码的索引指针(用户在地址簿中翻页时该指针必须刷新)，图形窗口位置和大小，所有 GUI 组件的句柄。为了能够正确地管理全局数据，地址簿 GUI 的行为控制主要实现以下几个方面：

- 发布 GUI：由于 GUI 需要在其他 MATLAB 7.0 任务运行的同时执行，所以 GUI 需要设置成不可中断、非模态的；
- 将一个地址簿装入阅读器中：有两种方式可以将地址簿 MAT 文件装入 GUI 中：发布 GUI 时使用参数指定 MAT 文件，如果用户不使用参数，则 MATLAB 7.0 装入默认的地址簿文件 addrbook.mat；用户使用 File 菜单的 Open 选项来装载其他 MAT 文件；
- 管理菜单回调行为：选择 Open 菜单项后将显示一个对话框（uigetfile），使用户能够浏览文件。对话框将返回文件名及其路径，并将返回值传递给 fullfile，从而确保路径对于任意的操作系统而言是合理构建的；Save 和 Save as 菜单用来保存对 MAT 文件的修改；Creat New 菜单将简单地清除 Contact Name 和 Contact Phone#编辑框的内容以便添加新的名字和电话号码。在添加新条目后，用户使用 Save 或 Save as 菜单项来保存地址簿。
- 管理按钮组件回调行为：Contact Name 编辑框显示地址簿条目的姓名，如果用户键入新的姓名，若当前地址簿中已存在该姓名，相应的电话将被显示；如果该姓名不存在，则显示一个询问对话框（questdlg）确认是否创建一个新的条目，如果不创建则返回到先前显示的姓名；Contact Phone#编辑框显示与 Contact Name 编辑框条目相匹配的电话号码。如果键入一个新的数字并回车，回调函数将发布一个询问对话框确认是否要对已存在的号码进行修改。按钮控件 Prev 和 Next 用来实现地址簿条目间的翻页。
- 地址簿将自定义重画函数，允许用户增大窗口的宽度以适应长姓名和电话号码，但是不允许缩小窗口宽度或改变窗口高度（这一限制不会限制 GUI 的使用，并且简化了必须保持窗口尺寸与组件大小相匹配的重画函数代码）。用户重画窗口并释放鼠标后将执行重画函数，重画后的窗口尺寸将被保存下来。

下面具体介绍解决方法：

步骤一：发布 GUI。用户可以不使用参数来调用应用程序 M 文件，在这种情况下 GUI 使用默认的地址簿 MAT 文件。用户也可以在参数中指定一个 MAT 文件，这就需要修改默认的应用程序 M 文件 GUI 初始化代码。修改后的初始化代码如下：

```
function varargout = address_book(varargin)
if nargin <= 1
%GUI 发布
fig = openfig(mfilename,'reuse');
set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
handles = guihandles(fig);
guidata(fig,handles);
if nargin ==0
    %装入默认的地址簿
    Check_And_Load([],handles)
elseif exist(varagin{1},'file')
    Check_And_Load(varargin{1},handles)
```

```
    else
        %如果文件不存在，则返回一个错误对话框并将文本设置为空字符串
        errordlg('File Not Found','File Load Error')
        set(handles.Contact_Name,'String','')
        set (handles.Contact_Phone,'String','')
    end
    if nargout >0
        varargout{1} = fig;
    end
    elseif ischar(varargin{1})
        %调用指定的子函数或回调函数
        try
            [varargout{1:nargout}] = feval(varargin{:});
        catch
        disp(lasterr);
        end
    end
```

步骤二：将一个地址簿装入阅读器中。首先要确认 MAT 文件。作为一个有效的地址簿文件，MAT 文件必须包含一个称作 Addresses 的结构体，该结构体有两个域，分别称为 Name 和 Phone。Check_And_Load 子函数将按照以下步骤来确认并装载这些数据。装载指定的或默认的文件；判断 MAT 文件是否为一个有效的地址簿文件，如果有效则显示数据，否则显示错误对话框；对于有效的 MAT 文件，返回 1，否则返回 0（该返回值由 Open 菜单项回调函数使用）；在句柄结构中保存条目 MAT 文件名和 Addresses 结构体；指示当前所显示的名字和地址的索引指针。

Check_And_Load 设置的代码如下：

```
    function pass = Check_And_Load(file,handles)
    % 初始化变量 pass 以判断文件是否有效
    pass = 0;
    %如果不指定文件名则使用默认名称，否则如果指定文件存在就装载它
    if isempty(file)
        file = 'addrbook.mat';
        handles.LastFile = file;
        guidata(handles.Address_Book,handles)
    end
    if exist(file) == 2
        data = load(file);
    end

    %判断 MAT 文件是否有效，当存在一个名位 Addresses 的变量
    %并且其两个域名为 Name 和 Phone 时，该文件有效
```

```
flds = fieldnames(data);
if (length(flds) == 1) & (strcmp(flds{1},'Addresses'))
    fields = fieldnames(data.Addresses);
    if (length(fields) == 2) &(strcmp(fields{1},'Name')) & (strcmp(fields{2},'Phone'))
        pass = 1;
    end
end
%如果文件有效则显示
if pass
    %给句柄结构体添加地址
    handles.Addresses = data.Addresses;
    %显示第一个条目
    set(handles.Contact_Name,'String',data.Addresses(1).Name)
    set(handles.Contact_Phone,'String',data.Addresses(1).Phone)
    %将索引指针设置为 1 并保存句柄
    handles.Index = 1;
    guidata(handles.Address_Book,handles)
else
    errordlg('Not a valid Address Book','Address Book Error')
end
```

步骤三：控制菜单回调行为。

首先看 Open 菜单项回调函数 Check_And_Load，该函数将确认并装载新的地址簿。

Open 菜单项的回调函数代码如下：

```
function varargout = Open_Callback(h, eventdata, handles, varargin)
[filename, pathname] = uigetfile( ...
    {'*.mat', 'All MAT-Files (*.mat)'; ...
        '*.*','All Files (*.*)'}, ...
    'Select Address Book');
%选择 Cancel 则返回
if isequal([filename,pathname],[0,0])
    return
    %否则构造文件全路径并检查装载这些文件
else
    File = fullfile(pathname,filename);
    %如果 MAT 文件无效，不保存信息
    if Check_And_Load(File,handles)
        handles.LastFIle = File;
        guidata(h,handles)
    end
end
```

再来看 Save 和 Save as 回调函数，该回调函数使用菜单项的 Tag 属性来判断究竟时 Save 还是 Save as 为回调对象，也就是说哪一个对象的句柄作为回调函数的第一个参数。如果用户选择的是 Save 菜单项，则调用 save 命令来保存 MAT 文件的新名字和电话号码；如果选择的是 Save as，则显示一个指定保存目标文件名的对话框，用户可以选择一个已有的文件，也可以输入新的文件名，对话框将返回文件名和路径。

- 使用 fullfile 来创建一个与平台武官的路径名；
- 调用 save 来保存 MAT 文件中的新数据；
- 刷新句柄结构体以包含新的 MAT 文件名；
- 调用 guidata 保存句柄结构体。

Save_Callback 回调函数代码如下：

```
function varargout = Save_Callback(h, eventdata, handles, varargin)
%获取被选菜单的 Tag 属性
Tag = get(h,'Tag');
%获取 address 数组
Addresses = handles.Addresses;
%根据所选项执行相应的行为
switch Tag
case 'Save'
    %保存到默认的地址簿文件中
    File = handles.LastFile;
    save(File,'Addresses')
case 'Save_As'
    %允许用户选择要保存的目标文件名
    [filename, pathname] = uiputfile( ...
        {'*.mat';'*.*'}, ...
        'Save as');
    %如果选择了 Cancel 则返回
    if isequal([filename,pathname],[0,0])
        return
    else
        %构造全路径名并保存
        File = fullfile(pathname,filename);
        save(File,'Addresses')
        handles.LastFile = File;
        guidata(h,handles)
    end
end
```

第三步设置 Creat New 菜单。Creat New 菜单的回调函数主要将编辑框的 String 属性设置为空。其代码如下：

```
function varargout = New_Callback(h, eventdata, handles, varargin)
```

```
set(handles.Contact_Name,'String','')
set(handles.Contact_Phone,'String','')
```

步骤四：控制组件回调行为。

首先看 Contact Name 编辑框回调函数。回调函数使用句柄结构体来访问地址簿的内容并获得索引指针，这样回调函数就能够判断用户修改之前编辑框中显示的是什么姓名。索引指针表明当前显示的姓名位置，在发布 GUI 时由 Check_And_Load 函数来添加地址簿和索引指针。如果用户添加了一个新的条目，回调函数将新的姓名添加到地址簿中并刷新索引指针来显示新的数值。刷新后的地址簿和索引指针再一次被保存在句柄结构体中。Contact Name 编辑框的回调函数代码如下：

```
function varargout = Contact_Name_Callback(h, eventdata, handles, varargin)
%在 Contact Name 和 Phone 编辑框众获取字符串
Current_Name = get(handles.Contact_Name,'string');
Current_Phone = get(handles.Contact_Phone,'string');
%如果为空则返回
if isempty(Current_Name)
    return
end
%从句柄结构体中获取当前地址列表
Addresses = handles.Addresses;
%搜索姓名列表，判断是否与一个已有姓名相同
for i = 1:length(Addresses)
    if strcmp(Addresses(i).Name,Current_Name)
        set(handles.Contact_Name,'string',Addresses(i).Name)
        set(handles.Contact_Phone,'string',Addresses(i).Phone)
        handles.Index = i;
        guidata(h,handles)
        return
    end
end
%如果是个新的姓名，请求创建一个新的条目
Answer=questdlg('Do you want to create a new entry?', ...
    'Create New Entry', ...
    'Yes','Cancel','Yes');
switch Answer
case 'Yes'
    Addresses(end+1).Name = Current_Name; % Grow array by 1
    Addresses(end).Phone = Current_Phone;
    index = length(Addresses);
    handles.Addresses = Addresses;
    handles.Index = index;
```

```
        guidata(h,handles)
        return
    case 'Cancel'
        %恢复为初始数值
        set(handles.Contact_Name,'string',Addresses(handles.Index).Name)
        set(handles.Contact_Phone,'String',Addresses(handles.Index).Phone)
        return
    end
```

再来看 Contact Phone #回调函数。和 Contact Name 编辑框类似，这个回调函数使用索引指针来刷新地址簿中的新数值或恢复为上一次的显示内容。所有当前地址簿和索引指针都保存在句柄结构体中，因而其他回调函数可以获得这些数据。以下是 Contact Phone #编辑框的回调函数代码：

```
    function varargout = Contact_Phone_Callback(h, eventdata, handles, varargin)
    Current_Phone = get(handles.Contact_Phone,'string');
    %如果有一个为空则返回
    if isempty(Current_Phone)
        return
    end
    %在句柄结构体众获取当前地址表
    Addresses = handles.Addresses;
    Answer=questdlg('Do you want to change the phone number?', ...
        'Change Phone Number', ...
        'Yes','Cancel','Yes');
    switch Answer
    case 'Yes'
        % If no name match was found create a new contact
        Addresses(handles.Index).Phone = Current_Phone;
        handles.Addresses = Addresses;
        guidata(h,handles)
        return
    case 'Cancel'
        %恢复为初始值
        set(handles.Contact_Phone,'String',Addresses(handles.Index).Phone)
        return
    en
```

最后看“Prev”和“Next” 按钮的回调函数。回调函数定义一个额外的参数 str 来表明被单击的按钮是“Prev”还是“Next”。“Prev”按钮的回调字符串以“Prev”为最后一个参数，而“Next”按钮则以“Next”为最后一个参数。在 case 语句中用 str 的数值来区分并实现每一个按钮的功能。Prev_Next_Callback 回调函数将从句柄结构体中获取当前的指针和地址，并且根据被选择的按钮来增加或减小索引指针的数值，并显示相应的名字和电话号码，最后

将在句柄结构体中存储新的索引指针数值并使用 guidata 来保存刷新后的结构体。

Prev_Next_Callback 回调函数代码如下：

```
function varargout = Prev_Next_Callback(h, eventdata, handles, str)
%获取索引指针和地址
index = handles.Index;
Addresses = handles.Addresses;
%根据被单击的按钮修改显示结果
switch str
case 'Prev'
    %索引值减 1
    i = index - 1;
    %如果索引值小于 1，设置其为 Addresses 数组最后一个元素的索引
    if i < 1
        i = length(Addresses);
    end
case 'Next'
    %索引值增加 1
    i = index + 1;

    %如果索引值超出范围，设置其为 Addresses 数组第一个元素的索引
    if i > length(Addresses)
        i = 1;
    end
end
%为被选索引获取相应数据
Current_Name = Addresses(i).Name;
Current_Phone = Addresses(i).Phone;
set(handles.Contact_Name,'string',Current_Name)
set(handles.Contact_Phone,'string',Current_Phone)
%刷新索引指针以反映新的索引
handles.Index = i;
guidata(h,handles)
```

步骤五：自定义重画函数。重画函数将对以下几种可能发生的情况进行处理。

- 改变宽度：如果新的宽度大于固有宽度，图形窗口将被设置为新的宽度。Contact Name 编辑框的尺寸随着窗口宽度的变化而变化。改变编辑框的宽度需要修改编辑框的 Units 为 normalized，重新设置编辑框的宽度为窗口宽度的 78.9%，将 Units 重新设置为 characters。如果新的宽度小于固有宽度，窗口被设置为固有宽度；
- 改变高度：如果用户视图修改窗口高度，高度始终被设置为固有高度。但是由于用户释放鼠标时会调用重画函数，所以重画函数不能总判断 GUI 在屏幕中的固有位置。因此，重画函数对窗口垂直分量作鼠标释放时的垂直位置 + 鼠标释放时的高度 − 原始高

度的设置。

当窗口从底部被重画时，窗口将保持原位；从顶部重画时，窗口将移动到鼠标释放处的位置。

重画函数调用 movegui 函数来确保无论用户在何处释放鼠标，重画后的窗口始终保持在屏幕中。第一次发布 GUI 时，GUI 窗口的大小和位置由 Position 属性决定，用户可以使用属性检查器来设置这个属性。

重画函数代码如下：

```
function ResizeFcn(hObject, eventdata, handles)
%获取窗口尺寸和位置
Figure_Size = get(hObject, 'Position');
%设置窗口固有尺寸
Original_Size = [ 0 0 94 19.230769230769234];
%如果重画窗口小于固有窗口尺寸，实行补偿
% original figure size then compensate
if (Figure_Size(3)<Original_Size(3)) | (Figure_Size(4) ~= Original_Size(4))
    if Figure_Size(3) < Original_Size(3)
        %如果宽度过小则设置为固有宽度
        set(hObject, 'Position',...
    [Figure_Size(1) Figure_Size(2) Original_Size(3) Original_Size(4)])
        Figure_Size = get(hObject, 'Position');
    end
    if Figure_Size(4) ~= Original_Size(4)
        %不允许修改高度
        set(hObject, 'Position',...
    [Figure_Size(1), Figure_Size(2)+Figure_Size(4)-Original_Size(4),...
    Figure_Size(3), Original_Size(4)])
    end
end
%设置 Contact Name 编辑框 Units 属性为 Normalized
set(handles.Contact_Name,'units','normalized')
%获取位置
C_N_pos = get(handles.Contact_Name,'Position');
%重新设置宽度使之与窗口相匹配
set(handles.Contact_Name,'Position',...
    [C_N_pos(1) C_N_pos(2)   0.789 C_N_pos(4)])
%将 units 重新设置为 Characters
set(handles.Contact_Name,'units','characters')
%在屏幕中重新设置 GUI
movegui(hObject, 'onscreen')
```

第 13 章　MATLAB 7.0 与 Word、Excel 的混合使用

MathWorks 公司开发的 MATLAB 7.0 Notebook 成功地将 Microsoft Word 和 MATLAB 7.0 结合在一起，为文字处理、科学计算和工程设计营造了一个完美的工作环境。这样 MATLAB 7.0 不仅兼具原有的计算能力，而且又增加了 Word 软件的编辑能力，MATLAB 7.0 Notebook 可以在 Word 中随时修改计算命令，随时计算并生成图像返回，使用户能在 Word 环境中“随心所欲地享用”MATLAB 7.0 的浩瀚科技资源。在 MATLAB 7.0 Notebook 中，用户可以在 word 文档中创建命令，然后送到 MATLAB 7.0 的后台中执行，最后将结果返回到 word 中。

建议撰写科技报告、论文、专著的科学工作者使用 MATLAB 7.0 Notebook，建议讲授、编写理工科教材的教师使用 MATLAB 7.0 Notebook，建议对于演算理工科习题的广大学生使用 MATLAB 7.0 Notebook。MATLAB 7.0 Notebook 的强大功能将会使你事半功倍。

除此之外，MATLAB 7.0 与 Excel 的混合使用使得用户在很多方面获得了极大的方便，在本章的第 2 节中将详细地介绍它们的使用方法。

13.1　Notebook 的安装和使用环境

从上面的介绍中可以看出，使用 MATLAB 7.0 Notebook 时，计算机中必须有 Word 和 MATLAB 7.0。本书以 MATLAB 7.0 和 Word 2000 为例。MATLAB 7.0 Notebook 文件又称为 M-book 文件，它是在 MATLAB 7.0 环境下安装的。下面介绍其具体步骤。

（1）在系统中分别安装 MATLAB 7.0 和 Word 2000，并启动 MATLAB 7.0 命令窗口；

（2）在命令窗口中输入：

```
>> notebook -setup
```

就会得到如下代码：

```
Welcome to the utility for setting up the MATLAB Notebook
for interfacing MATLAB to Microsoft Word
Choose your version of Microsoft Word:

[1] Microsoft Word 97

[2] Microsoft Word 2000

[3] Microsoft Word 2002 (XP)
```

```
[4] Exit, making no changes
```

（3）根据安装的 Word 的版本选择相应的代号，本文选择如下：

```
Microsoft Wor d Version: 2


Notebook setup is complete.
```

这样你就可以使用 MATLAB 7.0 Notebook 了。通过下面地两种方法可以打开一个 M-book 文件。

（1）直接在 MATLAB 7.0 命令窗口键入命令来新建或打开一个 M-book 文件：

```
>> notebook    %新建一个 M-book
>> notebook c:\documents\mymbook.doc %打开一个已经存在的 M-book
```

（2）直接打开 M-book.dot 文件，如图 13-1 所示。

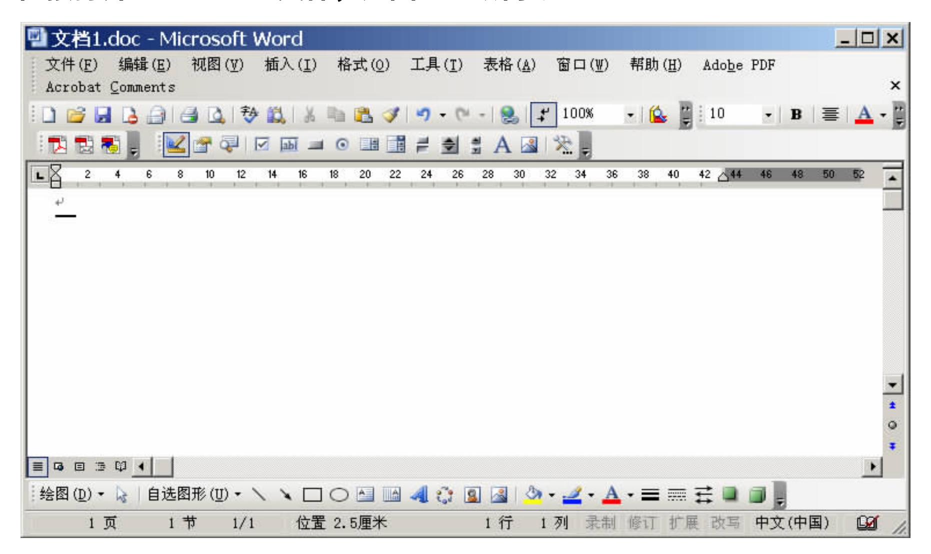


图 13-1　打开的 M-book.dot 文件

13.2　一个 Notebook 实例

使用 Notebook，在某种意义上就是在 Word 中使用 M-book 模板文档。Notebook 是通过动态链接和 MATLAB 7.0 交互的。Notebook 和 MATLAB 7.0 交互的基本单位为细胞（在 Notebook 中，Word 和 MATLAB 7.0 之间交换的信息叫做“细胞”）。M-book 需要把在 Word 中输入的 MATLAB 7.0 命令或者语句行组成细胞，再传到 MATLAB 7.0 中运行，运行输出的结果再以细胞的方式传回 Notebook。

1．在 Word 中执行命令的基本过程

Notebook 采用输入细胞(input cell)来定义 MATLAB 7.0 的输入命令。具体操作步骤如下：

- 采用文本格式输入命令，在命令结束时不要按回车和空格键；
- 从 Notebook 菜单中选中“Define Input Cell”选项，用来定义输入细胞；

- 从 Notebook 菜单中选中“Evaluate Cell”选项或者按 Ctrl+Enter 键。

其中输入细胞都显示为黑方括号包括的绿色字符，输出细胞都是黑方括号包括的蓝色字符，如果出现错误黑方括号包括的红色字符，其他文本都默认为黑色字符。如下：（作者注释：无专门标示的行为蓝色）

m=eye(3)　（作者注释：此行为绿色）

```
m =
        1       0       0
        0       1       0
        0       0       1
m/0             （作者注释：此行为绿色）
Warning: Divide by zero.
(Type "warning off MATLAB 7.0:divideByZero" to suppress this warning.)
ans =
    Inf     NaN     NaN
    NaN     Inf     NaN
    NaN     NaN     Inf
m=eye(3) n=eye(4)  （作者注释：此行为绿色）
??? format compact;m=eye(3) n=eye(4)
                                        |              （作者注释：此行以及上下两行为红色）
Error: Missing operator, comma, or semicolon.    （此行为绿色）
```

2．实例讲解

【例 13.1】在一段文本中间执行代码。

在 MATLAB 7.0 中，可以把输入细胞放在文本中间运行，而不影响其他文本。步骤如下：

- 输入样本如下，将 m=eye(3)放到文本中间。

```
m=eye(3)
```

- 选中命令，代码设置如下：

```
m=eye(3)
```

- 从 Notebook 菜单中选中“Evaluate Cell”选项或者按 Ctrl+Enter 键。将会出现上面例子中类似的结果。
- 如果要将输出细胞转化为普通文本，选中要转换的细胞，然后从 Notebook 菜单中选中“Undefine Cells”选项或者按 Alt+U 键。执行结果为：

```
m=eye(3)
m =
            1       0       0
            0       1       0
            0       0       1
```

【例 13.2】绘制一幅图片。

生成完整图形的多条图形指令必须定义在同一细胞群中：

```
t=0:0.1:20;y=1-cos(t).*exp(-t/5);
```

```
Time=[0,20,20,0];
Amplitude=[0.95,0.95,1.05,1.05];
fill(Time,Amplitude,'g'),axis([0,20,0,2]);
xlabel('Time'),ylabel('Amplitude');
hold on
plot(t,y,'r','LineWidth',2)
hold off
ymax=min(y)
```

将以上程序写到 Word 当中，然后全部选上，然后选择“Notebook|Define Input Cell”,最后从 Notebook 菜单中选中“Evaluate Cell”选项，或者按 Ctrl+Enter 键，得到的结果如图 13-2 所示。

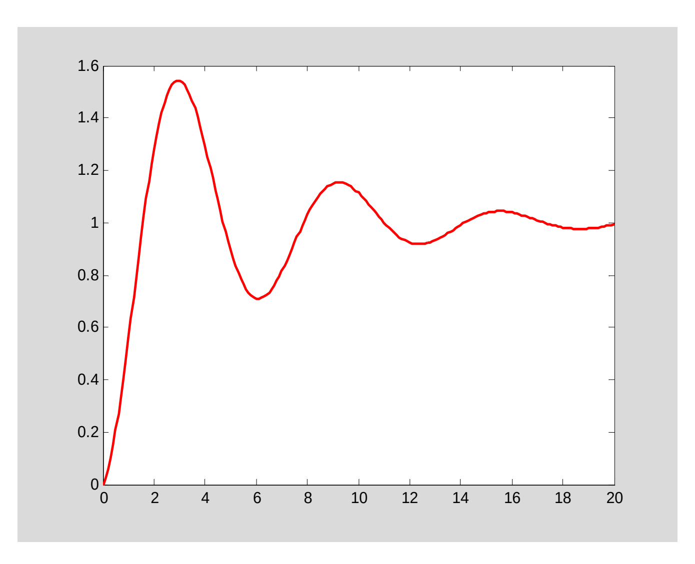


图 13-2　图片显示结果

13.3　Notebook 使用的几个问题

最后强调一下 Notebook 使用中应该特别注意的几个问题。

- M-book 文档即在 Word 中输入的 MATLAB 7.0 函数命令、语句或程序，其指令与标点符号都必须在英文状态下输入；
- 带鼠标操作的图形 MATLAB 7.0 交互指令最好不在 M-book 文档中运行；
- Windows 是一种多任务操作系统。但在运行 M-book 文档时，最好不运行其他程序，不执行其他任务，以免影响 M-book 文档中程序的正确执行；
- 由于计算机硬件与软件配合方面诸多不确定因素的影响，M-book 文档中的程序执行可能出现异常情况。这可以用以下方法解决：第一，MATLAB 7.0 主程序首行用语句 clear 以清除内存中的变量和函数；第二，重新启动计算机后，再执行 M-book 文档中的程序；第三，将 M-book 文档中的程序拷贝为 M 文件，然后到 MATLAB 7.0 的命令窗口

下去执行；

- MATLAB 7.0 指令在 M-book 文档中运行的速度比在 MATLAB 7.0 命令窗口中执行要慢很多，但这对撰写科技文章或数据没有什么影响；
- 可将细胞转换为普通文本。输入细胞在 Word 里编辑的文件中是绿色字符，而输出细胞是蓝色字符，这会影响 Word 文件的印刷效果。可以将 Notebook 中的输入细胞、输出细胞转换为普通文本，其转换步骤为先将光标放在细胞中，然后选中 Notebook 菜单中的“Undefine Cells”选项命令或者按组合键 Alt＋U。还可以用鼠标左键选中输入细胞即“刷黑”，接着运行 Notebook 菜单中的“Undefine Cells”子项命令，全部输入细胞和输出细胞都会被转换为文本格式显示，黑色的方括号对也被取消。所有绿色字符的输入细胞与蓝色字符的输出细胞均变成黑色字符。此时黑色字符为“Courier New”字体的 10 号字。在 Word 中，就可以按照撰写要求对其进行编辑；
- 如果需要的话，使用 Notebook 菜单中的“Bring MATLAB 7.0 to Font”命令或者按组合键 Alt＋M 可以把 MATLAB 7.0 的命令窗口调到前台；
- 使用 Notebook 菜单中的“Toogle Graph Output for Cell”命令可以控制是否显示输入细胞或输出细胞的输出图形。若控制为不输出图形，可将光标置于欲运行的细胞内，选中该“Toggle Graph Output for Cell”命令项，在细胞后将生成“no graph”，运行细胞就不输出图形；再一次选中该项，即可输出图形。

13.4　Excel link 的安装和使用环境

Excel link 是一个在 Microsoft Windows 环境下实现对 Microsoft Excel 和 MATLAB 7.0 进行链接的插件。通过对 Excel 和 MATLAB 7.0 的链接，用户可以在 Excel 的工作空间里，利用 Excel 的宏编程工具，使用 MATLAB 7.0 的数据处理和图像处理功能进行相关操作，同时由 Excel link 来保证两个工作环境中数据的交换和同步更新。使用 Excel link 时，不必脱离 Excel 环境，而是直接在 Excel 工作区或宏操作中调用 MATLAB 7.0 函数。Excel link 提供了 11 条功能函数来实现数据的链接和操作，对这些函数将在后面的小节中陆续说明。

13.4.1　Excel link 的安装

系统需要在 Windows 环境下先安装 Excel，然后再安装 MATLAB 7.0 和 Excel link。Excel link 的安装可以按 MATLAB 7.0 安装向导的提示进行，即在 MATLAB 7.0 安装组件选择框中选中 Excel link，并按提示继续安装。

安装完毕 Excel link 后，还需要在 Excel link 中做相应的设置，完成 Excel 与 Excel link 的链接。具体操作如下：

- 启动 Microsoft Excel，单击工具菜单（Tools），执行“加载宏”命令，如图 13-3 所示。



图 13-3 “加载宏”菜单

- 在打开的“加载宏”对话框中单击“浏览”按钮，选择用户自己的\matlab\toolbox\exlink 路径下的 excllink.xla 文件，然后单击“确定”按钮，如图 13-4 所示。

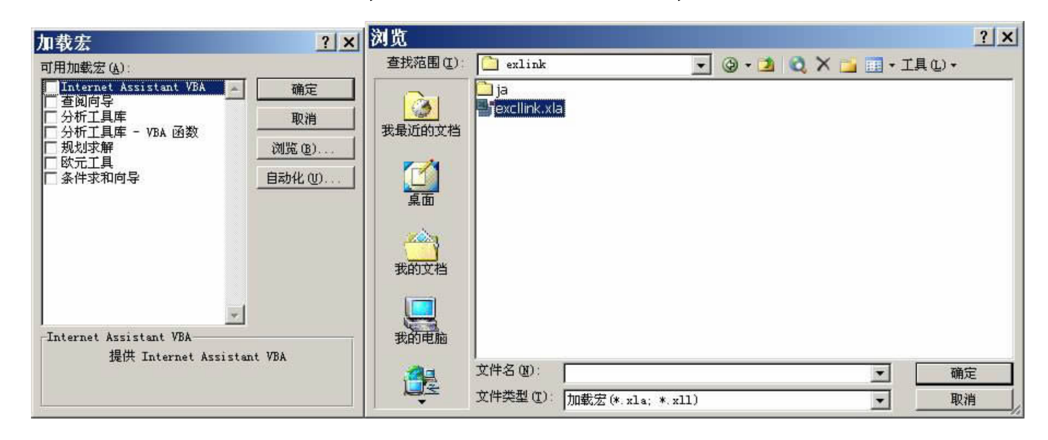


图 13-4 加载的宏文件

- 返回“加载宏”窗口，此时已经选中了“Excel link”选项，如图 13-5 所示。单击“确定”按钮后，Excel link 插件即可加载 MATLAB 7.0，并可以看到其运行窗口。

注意：这个 MATLAB 7.0 窗口是以传统方式打开的，也就是说没有启动 Java 界面。



图 13-5 已经选中 Excel link 界面

13.4.2　设置 Excel link 的启动方式

当用户按照上一节的步骤安装了 Excel link，并在 Excel 中进行了确认之后，在每一次启动 Excel 的时候，Excel link 和 MATLAB 7.0 将自动运行。如果不希望 Excel link 和 MATLAB 7.0 自动运行，那么可以通过在 Excel 数据表单元中输入“=MLAutoStart("no")”函数，即可改变 Excel 的初始化文件中对自动启动 Excel link 和 MATLAB 7.0 的设置，如图 13-6 所示。此时当再次启动该文件时，Excel link 和 MATLAB 7.0 就不会自动运行。

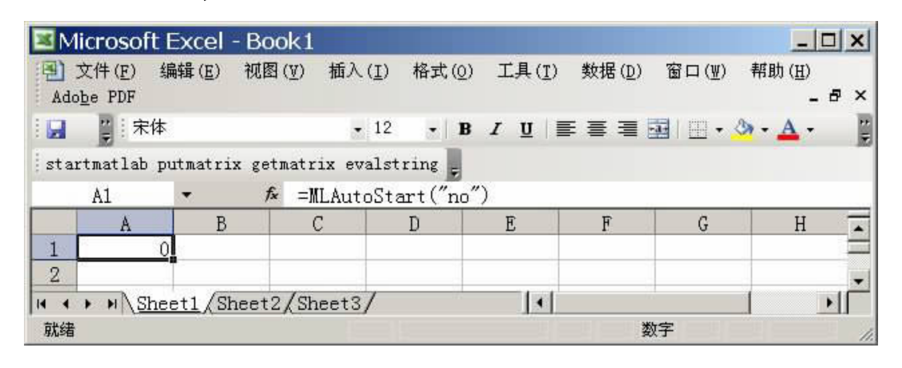


图 13-6　A1 单元中输入命令

用户也可以从 Excel 环境手动启动 Excel link 和 MATLAB 7.0。首先，在工具菜单中选择“宏”菜单，如图 13-7 所示。



图 13-7　选择宏

在打开的“宏”对话框中输入“MATLABinit”，如图 13-8 所示，单击“执行”按钮后即可启动 Excel link，并同时启动 MATLAB 7.0。

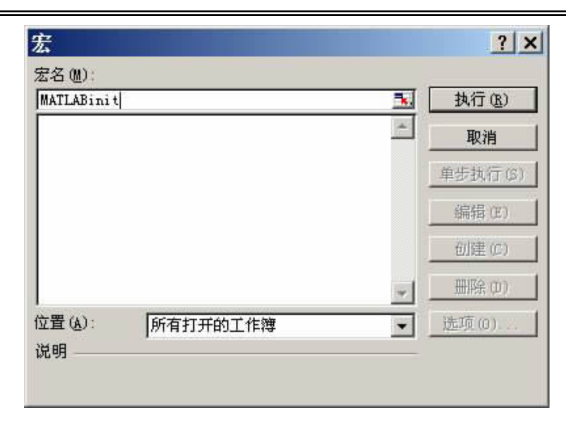


图 13-8　输入“ MATLAB 7.0init ”指令启动 Excel link 以及 MATLAB 7.0

13.4.3　终止 Excel link 的运行

当终止 Excel 的时候，Excel link 和 MATLAB 7.0 会被同时终止。

如果需要在 Excel 环境中终止 MATLAB 7.0 和 Excel link 的运行，则在工作表单元中输入“ =MLClose() ”。例如在 B1 单元中输入“ =MLClose() ”，如图 13-9 所示，确认后，将终止 MATLAB 7.0 和 Excel link 的运行，此时 B1 的状态如图 13-10 所示。

当需要重新启动 Excel link 和 MATLAB 7.0 时，可以选择用前面提到的 MLOpen 或 MATLABinit 命令来启动。

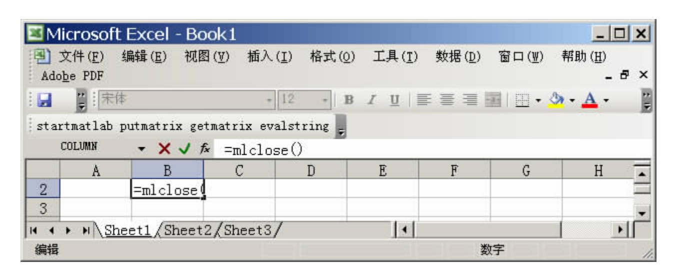


图 13-9　在 B1 单元中输入=mlclose()

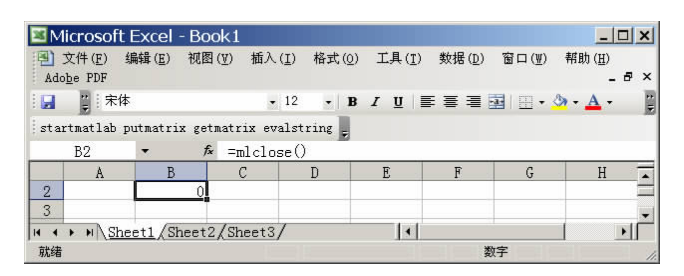


图 13-10　B1 的状态

13.5　一个 Excel link 实例

本节的应用举例中涉及到的 Excel 文件，包含在 MATLAB 7.0 安装路径下的 \toolbox\exlink 子目录下，文件名为：ExliSamp.xls。

在数据处理的过程中，使用回归和曲线拟合的目的是寻找最充分的反映变量之间的相互关系的函数，从效果而言，它们试图建立基于某一数据集的数学模型，MATLAB 7.0 提供了许多功能强大并且操作简便的矩阵操作和运算函数来支持此项业务。

本例中同时利用了回归分析和曲线拟合两种技术，并提供了工作表执行和宏命令执行两种方式。在 Excel 数据表中管理和显示数据，由 Excel link 函数把数据表中的数据拷贝到 MATLAB 7.0 中并执行 MATLAB 7.0 的计算和图形函数。宏命令执行模式实现了把运算结果返回给 Excel 数据表的功能。

13.5.1　数据表执行方式

启动 Excel、Excel link 和 MATLAB 7.0，打开示例文件 exlisamp.xls。

单击 exlisamp.xls 中的 Sheet1 标签，可以看到数据表中包含一个被命名为 DATA 的数组 A4：C28，如图 13-11 所示。

具体的执行步骤如下：

- 选中单元 E5，按 F2 键，回车执行 Excel link 函数 MLPutMatrix("data",DATA)，将 DATA 拷贝到 MATLAB 7.0 中。DATA 包含了对 3 个变量的 25 次观测值，并且已知观测值之间具有强线性相关性；
- 对 E8、E9 和 E10 执行相同的操作，在 Excel 中一次执行下列语句：

```
MLEvalString("y = data(:,3)")
MLEvalString("e = ones(length(data),1)")
MLEvalString("A = [e data(:,1:2)]")
```

这些语句实现对回归目标数据的设置，即用第一列和第二列数据对第三列进行回归。变量 y 用来存储第三列的数据，变量 A 的第一列为单位向量，后两列分别为原 data 的第一列和第二列。

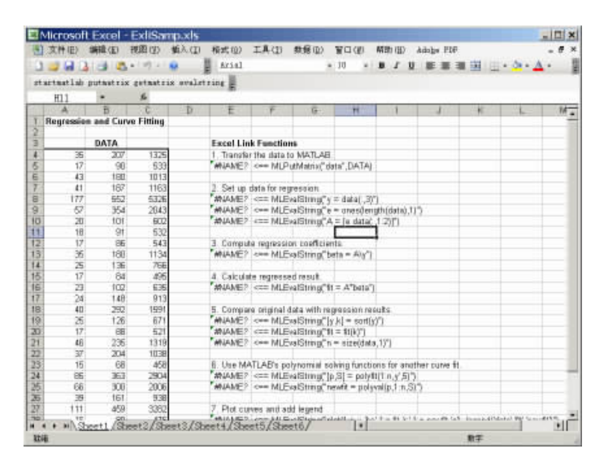


图 13-11　exliamp.xls 中的 sheet1 数据页

- **执行** E13 **的命令，代码设置如下：**

```
MLEvalString("beta = A\y")
```

利用 MATLAB 7.0 **中的反除法计算** A*beta = y **的回归系数。**

- **执行** E16 **的命令，代码设置如下：**

```
MLEvalString("fit = A*beta")
```

利用 MATLAB 7.0 **的矩阵向量乘法计算回归结果。**

- **执行** E19、E20 **和** E21 **的命令，代码设置如下：**

```
MLEvalString("[y,k] = sort(y)")
MLEvalString("fit = fit(k)")
MLEvalString("n = size(data,1)")
```

这些语句实现原始数据和回归数据的比较，将数据按升序排列后进行拟合，生成对观察数据的比例因子。

- **执行** E24 **和** E25 **的命令对** data **进行多项式拟合，代码设置如下：**

```
MLEvalString("[p,S] = polyfit(1:n,y',5)")
MLEvalString("newfit = polyval(p,1:n,S)")
```

- **执行** E28 **的命令，代码设置如下：**

```
MLEvalString("plot(1:n,y,'bo',1:n,fit,'r:',1:n,newfit,'g'); legend('data','fit','newfit')")
```

利用 MATLAB 7.0 **绘图函数显示原始数据** data、**拟合数据** fit **和多项式拟合数据** newfit，**如图** 13-12 **所示。**

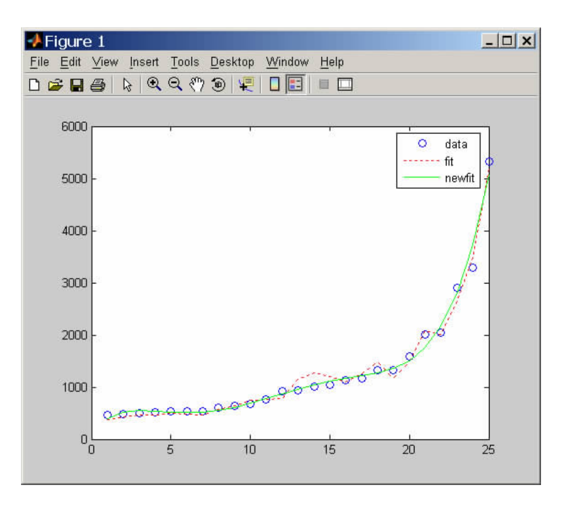


图 13-12 data、fit **和** newfit **的比较**

对图 13-12 **中** data、fit **和** newfit **三条曲线进行比较，可以发现数据具有很强的相关性，不是线性独立的，拟合曲线和原始数据并不是十分吻合，而** 5 **阶多项式拟合则显示了更加精确的数学模型。**

13.5.2 宏命令执行模式（Macro Version）

单击 Sheet2 **标签，可以运行本例的宏命令执行模式。**

激活单元 A4，**如图** 13-13 **所示，但先不要执行它。**

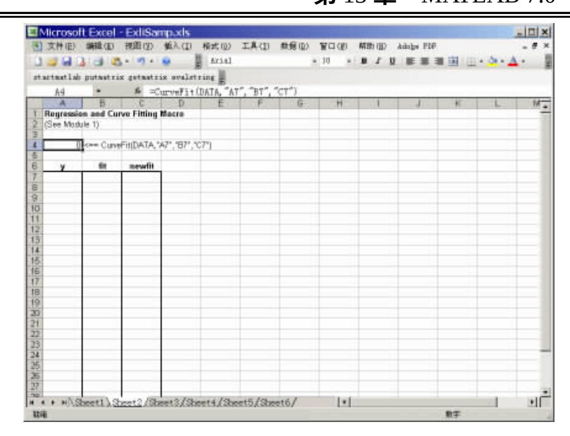


图 13-13　数据表 2

单元 A4 调用宏 CurveFit，可以在 Visual Basic 环境下打开 CurveFit。操作步骤如下：

- 在 Excel 中执行工具菜单下“宏”选项中的“Visual Basic 编辑器”选项，启动 Visual Basic；
- 在 Visual Basic 的工程里打开模块文件夹，如图 13-14 所示；

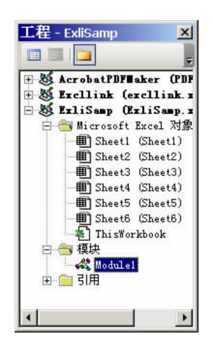


图 13-14　Visual Basic 的工程栏

- 选中 Moduel，则可以打开该模块，CurveFit 的程序代码如下：

```
Function CurveFit(aData, sTarget1, sTarget2, sTarget3)
    'MATLAB regression and curve fitting macro

    MLPutMatrix "data", aData
    MLEvalString "y = data(:,3)"
    MLEvalString "n = length(y)"
    MLEvalString "e = ones(n,1)"
    MLEvalString "A = [e data(:,1:2)]"
    MLEvalString "beta = A\y"
    MLEvalString "fit = A*beta"
    MLEvalString "[y,k] = sort(y)"
    MLEvalString "fit = fit(k)"
    MLEvalString "[p,S] = polyfit(1:n,y',5)"
    MLEvalString "newfit = polyval(p,1:n,S)"
```

```
        MLEvalString "plot(1:n,y,'bo',1:n,fit,'r:',1:n,newfit,'g');legend('data','fit','newfit')"
        MLGetMatrix "y", sTarget1
        MLGetMatrix "fit", sTarget2
        MLGetMatrix "newfit", sTarget3
    End Function
```

在此模块为打开的状态下，在工具菜单栏中单击“引用”命令，在弹出的“引用”对话框中查看是否选中了“Excellink”项，如图 13-15 所示，如果没有则需要加载。



图 13-15 “引用”对话框

返回到 Sheet2 的单元 A4，按 F2 回车，执行宏 CurveFit。宏 CurveFit 包含了工作表执行模式的第 1 步到第 7 步的操作，并且将排序后的数据 y、fit 和 newfit 从 MATLAB 7.0 拷贝到数据表中，如图 13-16 所示。

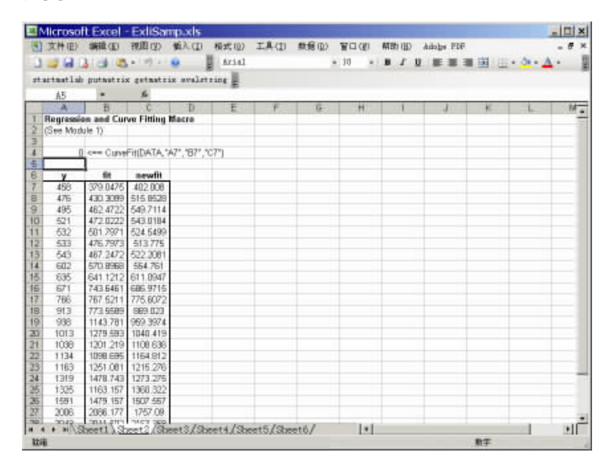


图 13-16 从 MATLAB 7.0 拷贝到数据表中的 y、fit 和 newfit

13.6 Excel link 使用的几个问题

为了更有效地利用 Excel link，下面对一些细节问题做进一步地说明。使用者必须牢记 Excel link 函数执行的是一个特定操作，而 Microsoft Excel 的函数返回一个确定的数值，所以

Excel 的操作和函数在 Excel link 函数的控制下会有不同的表现。

13.6.1　关于语法

1．大小写的区分

Excel link 函数名对字母的大小写不作区分，例如 MLPutMatrix 和 mlptmatrix 代表同一函数。而 MATLAB 7.0 函数名是区分大小写的，例如 BONDS、Bonds 和 bonds 就代表不同的变量。标准的 MATLAB 7.0 函数名通常使用小写字母，比如 plot()。

2．等式的起始标记

Excel 工作表等式通常以“＋”或“＝”作为起始标记，例如=mlputmatrix("a",C10)。

3．变量的定义方式

在大多数 Excel link 函数中有两种定义变量的方式，即直接定义和间接定义。将变量用双引号标记即可直接定义变量，例如 MLDeleteMatrix("Bonds")，函数中的不加双引号的工作区单元地址或行列名称被视为间接变量，函数对其指引内容进行操作。工作区单元地址可以包含页表序号。

13.6.2　关于工作表

1．工作表的单元值

Excel link 函数执行过程中其所在数据单元将一直显示其函数内容，函数执行完毕后，数据单元将被赋值为 0。

2．数据输入模式的是指

建议读者检查 Excel 工具菜单里“选项”的“编辑”选项卡中“按 Enter 键后移动”选项，如图 13-17 所示，以保证输入完毕经确认后再改变当前工作单元。

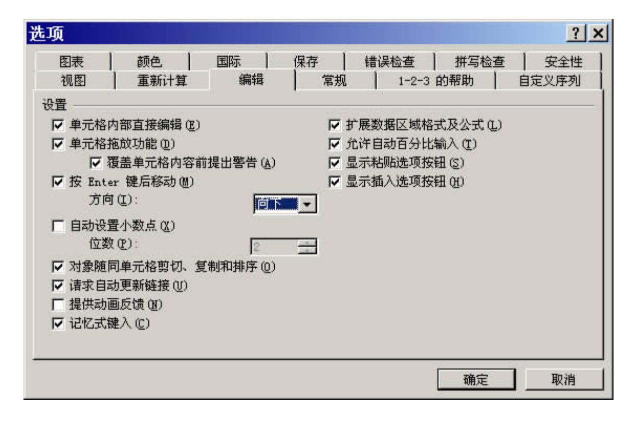


图 13-17　选择“按 Enter 键后移动”模式

第 14 章　编译工具箱

使用 MATLAB 7.0 的读者在感叹其运算功能强大的同时，往往希望程序可以运行的更快；希望获得可以摆脱 MATLAB 7.0 环境而独立运行的可执行软件。而 MATLAB 7.0 推出的编译器 Compiler 4.0 正是满足用户这些需求的软件。本章将重点向读者介绍 MATLAB 7.0 编译工具箱的使用。

14.1　编译器概述

MATLAB 7.0 Compiler 4.0 是将 M 文件作为它的输入，而产生可以重新分配的，并且是独立运行的应用程序或相应的软件组件。这些生成的应用程序以及软件组件都是与平台相关的。MATLAB 7.0 编译器可以产生下面几种应用程序或软件组件：

- 独立运行的程序：独立运行的程序顾名思义是在其运行的过程中可以不需要 MATLAB 7.0 软件的同时运行，甚至它们可以在没有安装 MATLAB 7.0 的机器上运行；
- C 和 C++共享库（在 Windows 操作系统中为动态链接库 DLL）：这些共享库可以在没有安装 MATLAB 7.0 的用户机器上运行；
- Excel 附件：需要 MATLAB 7.0 Builder；
- COM 对象：需要 MATLAB 7.0 Builder。

本章着重介绍前面两种类型的应用程序，而对于后面两种软件附件，在下面的章节进行相关介绍。

在进入正题之前，这里需要说明的是 MATLAB 7.0 编译器同 MATLAB 7.0 解释器之间的区别。MATLAB 7.0 解释器是指可以执行 MATLAB 7.0 命令、可执行的 M 文件以及 MEX 文件的应用程序，事实上，当我们使用 MATLAB 7.0 的时候，也就是在使用 MATLAB 7.0 的解释器。而 MATLAB 7.0 编译器是指将 M 文件作为其输入，同时生成独立的可执行文件或相关软件组件的程序，它可以由 mcc 命令调出。

14.2　编译器的安装和配置

14.2.1　配置 MATLAB 7.0 编译器的前提准备

MATLAB 7.0 在第一次使用其编译器时，将会自动对其进行适当的配置。然而，MATLAB 7.0 编译器的最初配置可以随时进行手动更改。下面将对此进行详细介绍。

1．安装 ANSI C/C++编译器。

在用户的计算机上，应事先安装以下任何一种与 MATLAB 7.0 适配的 ANSI C/C++编译器：

- Lcc C：由 MATLAB 7.0 自带，只是一个 C 编译器，不能用来编译 C++；
- Borland C++：版本为 5.3、5.4、5.5、5.6；
- Microsoft Visual C/C++ (MSVC)：版本为 6.0、7.0 和 7.1。

2．安装 MATLAB 7.0 Compiler。

MATLAB 7.0 Compiler 的安装过程一般包含在安装 MATLAB 7.0 之中，当用户选择 Typical 的安装模式时，MATLAB 7.0 Compiler 会被自动选为 MATLAB 7.0 的安装组件。当用户选择 Custom 安装模式时，在默认情况下，MATLAB 7.0 Compiler 选项是被选中的，如图 14-1 所示。

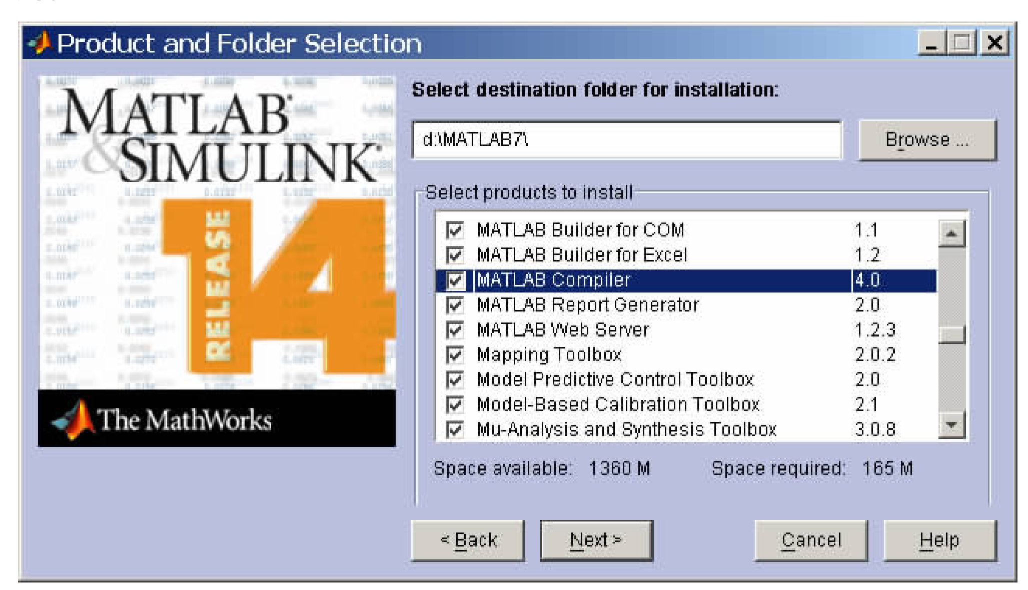


图 14-1　MATLAB 7.0 安装选项图

14.2.2　对编译器进行配置

本节将向读者介绍如何配置 C 或 C++编译器，使其可以与 MATLAB 7.0 编译器一起进行工作。在 MATLAB 7.0 中，函数命令 mbuild 可以大大地简化了配置 C 或 C++编译器的过程。一般而言，用户仅仅需要使用 mbuild 命令中的 setup 选项就可以轻松的设置好第三方编译器。

> 说明：在很多数的情况下，当用户将第三方编译器安装到其默认路径下时，是不需要运行 mbuild –setup 命令的。

当选用一个同 MATLAB 7.0 编译器相关联的 C 或 C++编译器时，可以使用命令 mbuild，同时加上参数-setup，代码设置如下：

```
mbuild –setup
```

命令执行结果代码如下：

```
Please choose your compiler for building standalone MATLAB
applications:
Would you like mbuild to locate installed compilers [y]/n? n
```

选择 n，从而手动选择编译器。

```
Select a compiler:
[1] Borland C++Builder version 6.0
[2] Borland C++Builder version 5.0
[3] Borland C++Builder version 4.0
[4] Borland C++Builder version 3.0
[5] Borland C/C++ version 5.02
[6] Borland C/C++ version 5.0
[7] Borland C/C++ (free command line tools) version 5.5
[8] Lcc C version 2.4
[9] Microsoft Visual C/C++ version 7.1
[10] Microsoft Visual C/C++ version 7.0
[11] Microsoft Visual C/C++ version 6.0

[0] None
Compiler: 11
```

这里选择 11，即 Microsoft Visual C/C++ version 6.0。

```
Your machine has a Microsoft Visual C/C++ compiler located at
D:\Microsoft Visual Studio. Do you want to use this
compiler [y]/n? y
```

选择 *y*，表示接受当前编译器的选择。

```
Please verify your choices:
Compiler: Microsoft Visual C/C++ 6.0
Location: D:\Applications\Microsoft Visual Studio
Are these correct?([y]/n): y
```

再次确认编译器的选择。

```
Try to update options file:
C:\WINNT\Profiles\username\Application
Data\MathWorks\MATLAB\R14\compopts.bat
From template:
\\sys\MATLAB\BIN\WIN32\mbuildopts\msvc60compp.bat
Done . . .    .
Updated
```

14.3　MATLAB 7.0 编译器的使用

14.3.1　编译过程

MATLAB 7.0 编译器 4.0 版本采用了 MATLAB 7.0 Component Runtime（MCR）技术，它是一组用来保证 M 文件执行的独立的共享库。MCR 提供了对 MATLAB 7.0 语言的完全支持。除此之外，MATLAB 7.0 编译器 4.0 版本还采用了 Component Technology File（CTF）存档来组织配置文件包。所有的 M 文件均采用了高级加密标准（AES）进行了密钥为 1024 位的加密，保存位 CTF 格式。每一个由 MATLAB 7.0 编译器生成的应用程序或者共享库均有一个与之相对应的 CTF 存档，其中包括了所有的基于 MATLAB 7.0 的 M 文件、MEX 文件等。

在 MATLAB 7.0 编译器中，生成独立文件或软件组件的过程是完全自动的。例如，为了生成一个独立运行的 MATLAB 7.0 应用程序，用户只需要提供一系列用来构成应用程序的 M 文件，然后编译器将会自动执行以下操作：

- 依赖性分析：分析判断输入的 M 文件、MEX 文件以及 P 文件所依赖的函数之间的关系，其中这些函数包括了输入文件所调用的 M 文件以及 MATLAB 7.0 提供的函数；
- 代码生成：生成所有用来生成目标组件的代码，其中包括：与从命令行中获得的 M 函数相关的 C 或 C++接口代码，对于共享库和组件来说，这些代码还包括了所有的接口函数；组件数据文件，其中包含了运行时执行 M 代码的相关信息，这些信息中有路径信息以及用来载入 CTF 存档中 M 代码的密钥；
- 存档生成 :在依赖型分析中生成的 MATLAB 7.0 可执行程序列表被用来生成 CTF 存档文件，其中包括程序运行时所需组件的数据。存档在加密后被压缩位一个单独的文件，同时路径信息也被保存；
- 编译：编译生成的 C 或 C++文件，得到目标代码。对于那些含有用户提供 C 或 C++代码的目标，这些代码同样会被编译；
- 链接：将生成的目标文件以及相关的 MATLAB 7.0 库链接起来，并生成最终的组件。

14.3.2　MCR 的安装

为了能够使用 MATLAB 7.0 编译器生成的组件，用户的机器上必须装有 MCR。下面介绍其具体的过程。

首先在 MATLAB 7.0 根目录下的\toolbox\compiler\deploy\win32 目录下，将 MCRInstaller.exe 拷贝到其他路径，然后双击进行安装，如图 14-2 所示。

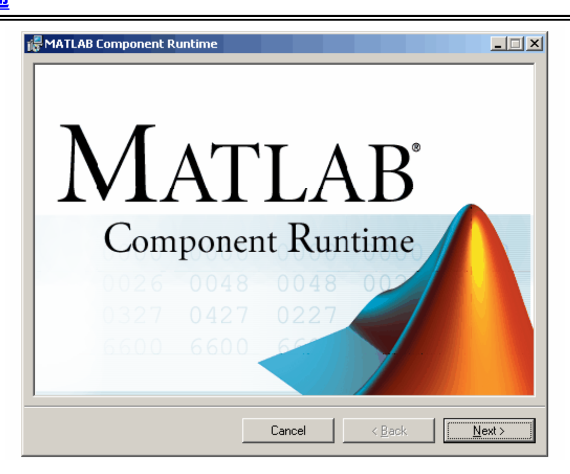


图 14-2　MCR 安装图示

然后选择安装路径，如图 14-3 所示。

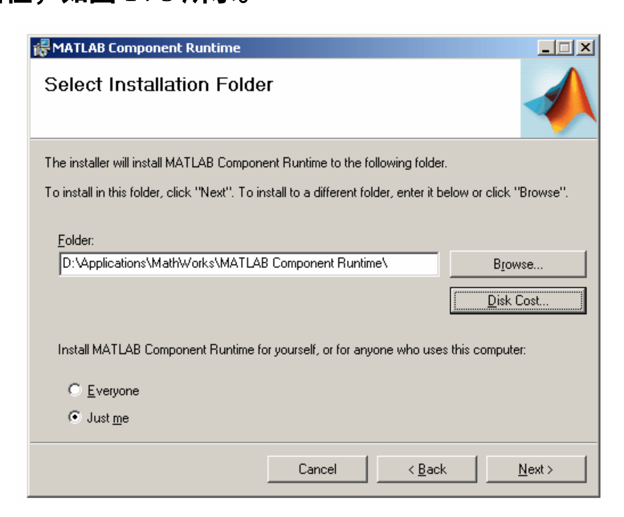


图 14-3　选择 MCR 安装路径

按照提示继续安装，直到提示安装结束。

14.3.3　编译指令 mcc

不管是想生成独立外部可执行程序，或者是想创建一个 C 共享库以及软件组件，只要源码是 M 文件，那么都可以借助编译命令 mcc 实现。用户可以在 MATLAB 7.0 环境中以及 DOS 或 UNIX 命令行环境中使用 mcc 指令。

用户可以声明一种或多种 MATLAB 7.0 编译器参数给 mcc 指令。大部分参数的名字由一个字母组成，用户可以独立地使用两个参数，如下面地示例代码：

```
mcc -m -g myfun
```

Macros 是 MathWorks 公式提供的用来简化编译任务编译参数，它使用户在面对一个简单问题的时候可以不再亲自手工组合多种参数来实现某种特定的编译过程，而是仅仅使用参数 -m 即可。

有了 Macros 参数，在处理大部分的编译问题时，我们均可以简单地键入下面格式的编译命令：

```
mcc -m myfun
```

14.3.4　创建独立的应用程序

为了说明编译的具体过程，下面举两个例子，分别代表不同类型的编译过程。

1．M 文件

建立一个脱离 MATLAB 7.0 环境，并可独立运行的外部程序。该程序的功能是对于给定矩阵 A，如果存在 S 使得 $S^{-1}AS = \Lambda$，则要求出一个 S，否则给出信息说明所给的矩阵不能对角化。

首先编写两个 M 函数文件，即 exm2.m 和 exm2_f.m（第一个文件是主文件）。

exm2.m 的代码如下：

```
function exm2
A = [4,0,0;0,3,1;0,1,3];
S = exm2_f(A)
```

exm2_f.m 的代码如下：

```
function S = exm2_f(A)
[m,n] = size(A);
if m ~= n
      error('输入矩阵应是方阵');
end
e = eig(A);
%检查输入矩阵的特征值是否各异
same = 0;
for i = 1:m-1
    for j = (i+1):m
          if e(j) == e(i)
              same = 1;
          end
    end
end
%A 可以对角化的条件是 A 具有各异特征值或者 A 位埃米尔特矩阵。
if any(any((A'-A)))&(same ==1)
       error('矩阵无法对角化');
end
[v,d] = eig(A);
S = v;
```

把这两个函数文件保存在在用户自己的目录，例如 d:\matlab7\work，并在 MATLAB 7.0

中运行检验，代码设置如下：

```
exm2
```

得到结果如下：

```
S =
          0          0     1.0000
    -0.7071     0.7071          0
     0.7071     0.7071          0
```

生成独立的外部可执行程序。在 MATLAB 7.0 指令窗口中，运行如下指令：

```
mcc -m exm2 exm2_f
```

打开 DOS 窗口，在 d:\matlab7\work 目录下，运行 exm2.exe，得到结果如图 14-4 所示。

```
C:\WINNT\system32\cmd.exe
Microsoft Windows 2000 [Version 5.00.2195]
(C) 版权所有 1985-2000 Microsoft Corp.

C:\Documents and Settings\Administrator>cd d:\matlab7\work

C:\Documents and Settings\Administrator>d:

D:\MATLAB7\work>exm2

S =

         0         0    1.0000
   -0.7071    0.7071         0
    0.7071    0.7071         0


D:\MATLAB7\work>
```

图 14-4　在 DOS 窗口运行生成程序 exm2.exe 所得的结果

2．M 文件和 C 文件混合

本例给出了如何在 C 语言文件中调用编译过的 M 函数。考虑一个由两部分组成的程序：

- multarg.m：其中包含一个函数 multarg；
- multargp.c：其中包含一个 C 语言函数 main。

multarg.m 中包含的 multarg 函数包含两个输入参数和两个输出参数。源代码如下：

```
function [a,b] = multarg(x,y)
a = (x + y) * pi;
b = svd(svd(a));
```

multargp.c 中的 main 函数调用 multarg.m 文件中的 multarg 函数，并显示 multarg 函数的返回值。源代码如下：

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "libMultpkg.h"
/*
```

```
 * Function prototype; the MATLAB 7.0 Compiler creates mlfMultarg
 *    from multarg.m
 */
void PrintHandler( const char *text )
{
    printf(text);
}
/* main 函数用来调用 multarg 函数  */
int main( )
{
#define ROWS    3
#define COLS    3
    mclOutputHandlerFcn PrintHandler;
    mxArray *a = NULL, *b = NULL, *x, *y;
    double    x_pr[ROWS * COLS] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    double    x_pi[ROWS * COLS] = {9, 2, 3, 4, 5, 6, 7, 8, 1};
    double    y_pr[ROWS * COLS] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    double    y_pi[ROWS * COLS] = {2, 9, 3, 4, 5, 6, 7, 1, 8};
    double *a_pr, *a_pi, value_of_scalar_b;

    /*  初始化图形句柄，同时声明显示格式
      */
    mclInitializeApplication(NULL,0);
    libMultpkgInitializeWithHandlers(PrintHandler, PrintHandler);


    /*  生成输入矩阵  "x" */
    x = mxCreateDoubleMatrix(ROWS, COLS, mxCOMPLEX);
    memcpy(mxGetPr(x), x_pr, ROWS * COLS * sizeof(double));
    memcpy(mxGetPi(x), x_pi, ROWS * COLS * sizeof(double));

    /*生成输入矩阵  "y" */
    y = mxCreateDoubleMatrix(ROWS, COLS, mxCOMPLEX);
    memcpy(mxGetPr(y), y_pr, ROWS * COLS * sizeof(double));
    memcpy(mxGetPi(y), y_pi, ROWS * COLS * sizeof(double));

    /*  调用 mlfMultarg 函数. */
    mlfMultarg(2, &a, &b, x, y);

    /*  显示得到的矩阵  "a". */
```

```
        mlfPrintmatrix(a);

        /*  显示输出矩阵 "b" */
        mlfPrintmatrix(b);

        /*  销毁中间矩阵变量. */
        mxDestroyArray(a);
        mxDestroyArray(b);
        libMultpkgTerminate();
        mclTerminateApplication();
        return(0);
}
```

将上述两个程序分别保存到用户工作目录下，在 MATLAB 7.0 的运行环境中，对两个文件进行编译链接。

```
mcc -W lib:libMultpkg -T link:exe multarg printmatrix multargp.c
```

可以得到运行结果如图 14-5 所示。

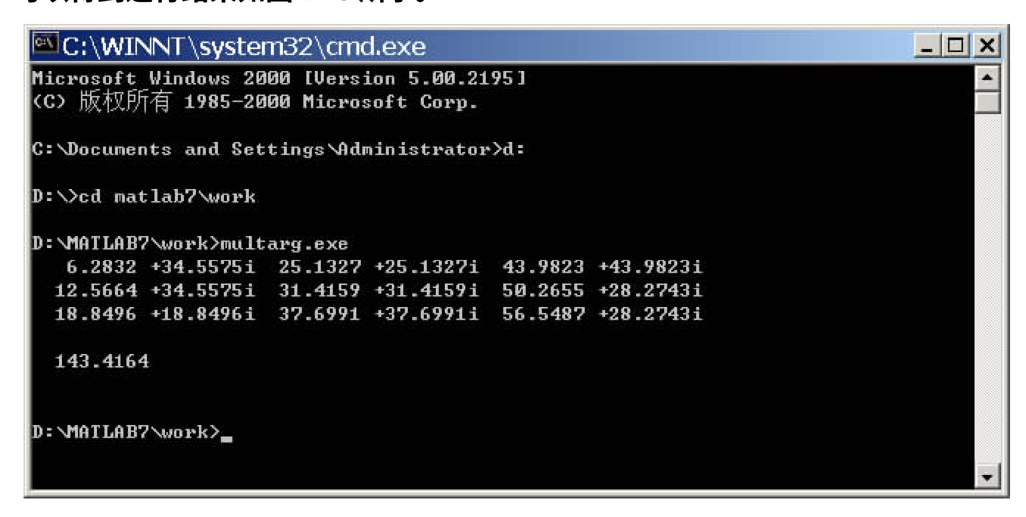


图 14-5　混合源码运行结果

3．包含绘图指令的 M 文件

由于安装了 MCR，因此在编译的过程中可以加入 MATLAB 7.0 图形库。

定义 M 文件 test_plot.m，源代码如下：

```
function test_plot
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
axis square; grid on
```

在 MATLAB 7.0 命令窗口中输入如下代码：

```
mcc -m test_plot.m
```

得到编译链接后的文件 test_plot.exe。双击执行，可以得到结果如图 14-6 所示。

图 14-6　带绘图指令的 M 文件编译运行结果

4．由含 feval 指令的 M 文件生成 EXE 文件

feval 指令的特殊之处在于它的第一个输入参量是函数名字符串。这里要讨论的是当 M 源文件包含该指令时，将如何从这个源文件获得 EXE 文件。在 MATLAB 7.0 中给出了编译注记（Pragmas）处理 feval 对函数的调用。其格式为：

```
%#function <function_name-list>
```

编译注记提示 MATLAB 7.0 编译器在编译的过程中需要将函数列表（function_name-list）中的函数均包含进去，而无论编译器的依赖性分析过程中是否定位到函数列表中的函数。当然，用户也不可将编译注记指向不存在的函数。

下面是一个采用编译注记的例子，由两个函数组成 mywindow.m 和 test1.m。其中 test1 用来调用 mywindow.m 文件。

mywindow.m 文件的内容为：

```
function ret = mywindow(data,fitlerName)
%对输入的数据进行加窗

%获得输入数据的长度
N= length(data);

%列出所有的窗函数
%#function bartlett, barthannwin, blackman,blackmanharris,bohmanwin,chebwin, flattopwin, gausswin, hamming,hann,kaiser, nuttallwin,parzenwin, rectwin, tukeywin,triang
window = feval(fitlerName,N);% 对输入的数据进行加窗.
```

```
ret = data.*window;
```

test1.m 文件的内容为：

```
function test1
data = sin((1:10)/10*pi)';
newdata = mywindow(data,'chebwin');
disp(newdata')
```

对主文件进行编译，可以产生符合要求的独立可执行程序 test1.exe。

```
mcc -m test1.m
```

程序执行结果如图 14-7 所示。

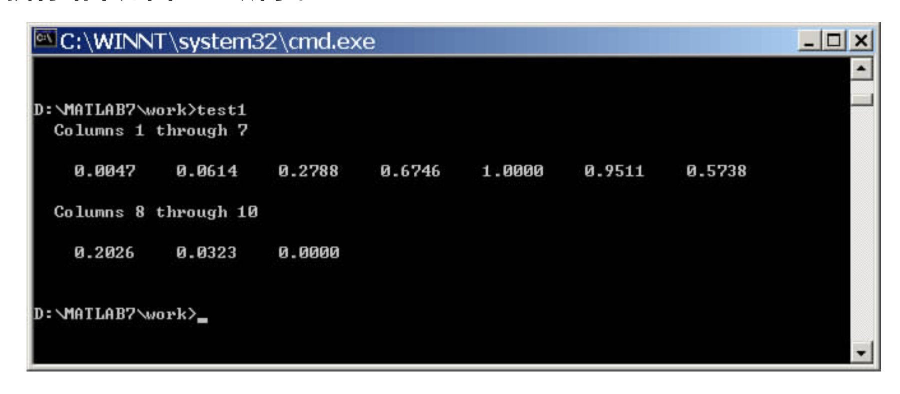


图 14-7　含 feval 指令的 M 文件编译运行结果

第 15 章　应用程序接口

MATLAB 虽然是一个完整的科学计算与可视化的环境，但在很多情况下它仍需与其他外部程序沟通，可用 MATLAB 提供的应用程序接口（Application Program Interface，简称 API）来实现此功能。

MATLAB 应用程序接口包括 3 部分内容，分别为 MEX 文件——外部程序调用接口，在 MATLAB 中调用 C 语言或者 Fortran 语言编写的程序，以提高数据处理的效率；MAT 文件应用程序——数据输入输出接口，向 MATLAB 环境传送数据或从 MATLAB 环境接收数据，即实现 MATLAB 系统与外部环境的数据交换；MATLAB 计算引擎函数库——在 MATLAB 和其他应用程序间建立客户机/服务器关系，将 MATLAB 作为一个计算引擎，在其他应用程序中调用，从而降低程序设计的工作量。MATLAB 在 Windows 平台上支持 Microsoft 提出的 COM 标准，同时支持 Java 语言，所以 MATLAB 几乎可以同 Windows 平台上任何一种软件或者开发语言进行交互。

本章将重点讨论这些应用程序接口的实现方法。

15.1　创建 C 语言 MEX 文件

本节将介绍在 MATLAB 中创建 C 语言的 MEX 文件的方法，在这之前，首先对 MEX 文件作一个简要的介绍。

15.1.1　MEX 文件简介

MEX 从字面上是 MATLAB 和 Executable 两个单词的缩写。一般 MEX 文件使用 C 语言或者 Fortran 语言进行开发，通过适当的编译后，生成的目标文件能够被 M 语言解释器调用执行，这种文件在 Microsoft Windows 下使用后缀.dll。MEX 文件重要应用于已存在较大规模的 C 和 Fortran 程序，可以比较容易的在 MATLAB 中调用或是在 MATLAB 中运行不很有效，存在计算瓶颈或是直接面向硬件编写的 C 和 Fortran 程序可以通过 MEX 文件被 MATLAB 调用。

“阵列”是 MATLAB 惟一能处理的对象，在 C 语言中，MATLAB“阵列”用结构体 mxArray 定义。

下面通过一个最简单的 MEX 文件示例来认识 MEX 源文件的结构。

【例 15.1】简单 MEX 文件示例——mexhello.c

示例代码设置如下：

```
/*necessary header file*/
#include "mex.h"
```

```
/*entrance function of MEX function*/
void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray *prhs[])
{
    /*content of function ,transfer other function*/
   mexPrintf("Hello world!");
}
```

这是标准的 C 语言源文件，其程序的基本语法完全是 ANSI C 的语法结构。首先是头文件的包含语句，所有的源文件中必须包含 mex.h，它完成了所有 C 语言 MEX 函数的原型声明，还包含了 matrix.h 文件，即对 mx 函数和数据类型的声明和定义。

接下来是 C 语言 MEX 源文件的入口函数部分，这也是必需的部分，并且书写形式固定。括号里的 4 个变量分别表示输入参数的个数、输入参数、输出参数的个数和输出参数。

需要注意的是，MATLAB 7.0 的 MEX 源文件中不能包括任何汉字，否则会出现编译连接错误。

上述 MEX 源文件编写完毕后，保存在 MATLAB 的当前工作路径下，名称为 mexhello.c，接着在命令窗口键入以下指令：

```
>> mex mexhello.c
```

此命令对 MEX 文件进行编译，创建出 MATLAB 的 MEX 文件，可以在 MATLAB 命令行中运行下面的语句：

```
>> mexhello
Hello world!
```

注意：上述编译过程也可以在命令行提示符窗口中完成。

需要注意的是 MATLAB 本身是一个高效的系统，编程简单，效率高，不到万不得已，建议不要使用 MEX 手段。

15.1.2 编写 C MEX 文件

编写 MEX 文件源程序时，要用到两类 API 库函数，即 mx-库函数和 mex-库函数，前者用于在 C 语言中创建、访问、操作和删除结构体 mxArray，后者用于与 MATLAB 环境进行交互。有关这些库函数的详细说明可参看 MATLAB 的 help 文件。

下面通过一个实例来演示 C MEX 的编写过程。

【例 15.2】生成随机数——执行 MATLAB 函数

示例代码设置如下：

```
/*MEX function example,create random numbers——randomnum.c*/
#include "mex.h"
/*MEX function entrance*/
    void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray *prhs[])
    {
        double *m,*n,*flag;
        /*error detecting*/
```

```
        if(nlhs>1||nrhs!=3)
    mexErrMsgTxt("error:wrong numbers of input/output!");
    m=mxGetPr(prhs[0]);
    n=mxGetPr(prhs[1]);
    flag=mxGetPr(prhs[2]);
    plhs[0]=mxCreateDoubleMatrix(*m,*n,mxREAL);
    /*set output flag*/
    mexSetTrapFlag(*flag);
    /*wrong transfer-->function rnd does not exsit*/
        mexCallMATLAB(1,plhs,2,prhs,"rnd");
        /*continue to run MEX function file so rand function is transfered by MEX file*/
        mexPrintf("user set Flag to 1.\n");
        /*rright transfer*/
        mexCallMATLAB(1,plhs,2,prhs,"rand");
        }
```

本例的源程序中使用 mexCallMATLAB 函数调用了 MATLAB 的随机数产生函数 rand。mexCallMATLAB 函数的定义如下：

```
int mexCallMATLAB(int nlhs,mxArray *plhs[],int nrhs,msArray *prhs[],const char *command_name);
```

其中前 4 个参数为需要输入到被调用函数中的输入、输出参数，而最后一个参数为被调用的函数名称。当执行 command_name 函数出现错误时，根据 mexSetTrapFlag 函数的设置进行错误处理，其定义如下：

```
void mexSetTrapFlag(int trap_flag);
```

若输入参数为 0，则将程序的控制权交给 MATLAB，退出 MEX 函数的执行，若输入参数为 1，则将程序的控制权交给 MEX 函数，继续执行 MEX 函数。例子中的第一次 MATLAB 调用是错误的，不存在 rnd 函数，程序如何执行将决定于 *flag* 值的设定。

编译并执行 randomnum.c，代码设置如下：

```
>> mex randomnum.c
>> y=randomnum(3,2,0)   %flag 为 0
??? Undefined command/function 'rnd'.
%调用不成功，再次运行，flag 设为 1
>> y=randomnum(3,2,1)
??? Undefined command/function 'rnd'.
user set Flag to 1.
y =
    0.9501      0.4860
    0.2311      0.8913
    0.6068      0.7621
```

不同的实例将会用到不同的 mex 函数，有关其他 mex 函数的使用，用户可以参看 help 说明和其他专门资料的例程。

此外，MEX 文件还可用 Visual Studio 来创建，适合习惯利用集成开发环境的程序员使用。

15.2 创建 Fortran 语言 MEX 文件

使用 Fortran 语言创建 MEX 文件的方法和过程与创建 C 语言 MEX 文件的方法和过程几乎一样，无非是使用的高级编程语言有所不同而已。Fortran 语言在科学计算方面的应用比 C 语言要广泛，它的数值稳定性和多种数值变量是 C 语言无法匹敌的。

本节主要介绍 Fortran 语言 MEX 文件的创建。

15.2.1 Fortran 语言 MEX 文件简介

以一个简单的例程介绍 Fortran 语言 MEX 文件。

【例 15.3】简单的 Fortran 语言 MEX 文件——mexhello.c

```
C    entrance function
subroutine mexFunction(nlhs,plhs,nrhs,prhs)
C    -----------------------------------------
C    parameters defination
integer plhs(*),prhs(*)
integer nlhs,nrhs
C    code
call mexPrintf('Hello MATLAB World!')
C    end of file
return
end
```

它的功能也是在 MATLAB 命令窗口输出一段文字。在 M 文件编辑器中输入完毕后，保存在当前工作目录下，名称为 mexhello.f。然后编译运行：

```
>> mex mexhello.f
>> mexhello
Hello MATLAB World!
```

> 注意：在进行编译之前，一定要在 MATLAB 环境下完成 MEX 文件编译器的配置，本书例程中所使用的 Fortran 开发环境是 Compaq Visual Fortran。

与 C MEX 类似，首先是入口函数的声明部分（没有了头文件的包含语句），输入参数和输出参数的含义与 C 相同，然后是源程序的主体部分，进行数据的存取和内存的分配，调用相应的计算子程序或者函数。程序的结尾是 Fortran 的特别要求，必须有 return 和 end 关键字作为结束。

MATLAB 的数据在 C 中使用 mxArray 数据类型来表示，但是在 Fortran 中没有显性的定义该数据结构，而是通过一种所谓的“指针”类型数据完成 Fortran 和 MATLAB 之间的数据传递。MATLAB 将需要传递的 mxArray 数据指针保存为一个整数类型的变量，例如在 mexFunction 入口函数中声明的 *prhs* 和 *plhs*，然后在 Fortran 程序中，通过能够访问指针的 Fortran 语言 mx 函数访问 mxArray 数据，获取其中的实际数据。

15.2.2　Fortran MEX 文件示例

【例 15.4】创建稀疏矩阵

```
C       fcreatesparse.f
C       create sparse array
C       entrance function
C       parameters declaration
integer plhs(*),prhs(*)
integer nlhs,brhs
C       pointer
integer ir,jc,pr
C       actual data
real*8 pdata(6)
data pdata/3,1,10,-7,2,-2/
integer irdata(6)
data irdata/0,3,3,1,2,3/
integer jcdata(5)
data jcdata/0,2,2,5,6/

integer mxGetIr,mxGetJc,mxGetPr
integer mxCreateSparse
integer mxCopyReal8ToPtr
integer mxCopyInteger4ToPtr
C       code
plhs(1)=mxCreateSparse(4,4,6,0)
ir=mxGetIr(plhs(1))
jc=mxGetJc(plhs(1))
pr=mxGetPr(plhs(1))
C       memory copy
call mxCopyReal8ToPtr(pdata,pr,6)
call mxCopyInteger4ToPtr(irdata,ir,6)
call mxCopyInteger4ToPtr(jcdata,jc,5)
C       end of file
return
end
```

本例的核心部分首先使用 mxCreateSparse 函数创建了稀疏矩阵的 mxArray 数据对象，然后分别使用 mxGetIr、mxGetJc、mxGetPr 3 个函数获取稀疏矩阵的 3 个重要指针 *ir*、*jc* 和 *pr*，这里是实数的矩阵，所以没有 *pi* 指针。接着使用 mxCopy 函数完成数据内存的复制。需要注意的是整数类型的数据拷贝，在 MATLAB 的 API 函数库中，有 3 个不同的函数用于整数类型数据的拷贝，分别是 mxCopyInteger1ToPtr、mxCopyInteger2ToPtr 和 mxCopyInteger4ToPtr。

同样，还有 3 个函数用于将指针数据拷贝给相应的整型数据，这里不再列举。具体程序编写过程中要针对具体的整数数据类型选取不同的拷贝函数。下面是编译运行该文件后的一些结果：

```
>> mex fcreatesparse.f
>>A=fcreatesparse
A=
   (1,1)          3
   (4,1)          1
   (1,3)          10
   (2,3)          -7
   (3,3)          2
   (4,4)          -2
>>full(A)
ans=
      3     0     10     0
      0     0     -7     0
      0     0     2      0
      1     0     0      -2
```

15.3　MAT 文件应用

MATLAB 与外界交互数据有直接输入输出法（用于数据量小的情况）、数据的 M 文件输入和 diary 文件输出法、输入输出数据的 ASCII 文件法、低层 IO 指令读写数据法、读写数据的 MEX 文件法和 MAT 文件法等多种方法。其中 MAT 文件法是先用 C 或者 Fortran 编写一个专门的文件，实现原数据与 MAT 文件格式之间的相互转换，进而借助 save，load 实现 MATLAB 对原数据的读写。MAT 文件是 MATLAB 数据存储的默认文件格式，由文件头和数据组成，文件扩展名是.mat。

【例 15.5】save 和 load 指令示例

在 MATLAB 命令行窗口中，键入下面的指令：

```
>> clear all
>> %create variables
>> x1=2;x2=3;x3=4;y1=0;
>> %save data
>> save xdata x1 x2
>> %check MAT file in current directory
>> dir *.mat
xdata.mat
>> clear all
>> %load data(binary)
```

```
>> load xdata
>> whos
  Name        Size                    Bytes   Class
  x1          1x1                       8     double array
  x2          1x1                       8     double array
Grand total is 2 elements using 16 bytes
```

下面将介绍 MAT 文件的应用，所谓 MAT 文件的应用就是指读写 MAT 数据文件的 C 语言或者 Fortran 语言应用程序，它可以是 MEX 文件也可以是独立的可执行应用程序。为了便于读写 MAT 文件，MATLAB 提供了响应的接口函数——mat 函数，MAT 文件应用程序就是利用这些 mat 函数完成 MAT 数据文件的读写工作。下面给出一个 C 语言 MAT 文件应用的例子，可从中了解 MAT 文件应用程序的基本结构和 MAT 文件应用的基本过程。

【例 15.6】在 C 语言 MEX 文件中创建 MAT 数据文件——mexcreatematfile.c

示例代码设置如下：

```
/*necessory header file*/
#include "mat.h"
#include "math.h"
/*MEX function file entrance*/
void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray *prhs[])
{
    mxArray *string,*scalar,*array;
    double img[]={0,1,2,3,4,5,6,7,8};
    double real[]={8,7,6,5,4,3,2,1,0};
    double *pr,*pi;
    mxChar *filename;
    MATFile *file;
    int buflen=0;
    /*judge the input/output*/
    if(nrhs!=1)
        mexErrMsgTxt("an input is necessory to be the filename!");
    if(mxGetClassID(prhs[0])!=mxCHAR_CLASS)
        mexErrMsgTxt("input must be characters, stand for filename!");
        /*create data*/
        string=mxCreateString("MATLAB is wonderful!");
        scalar=mxCreateDoubleScalar(2003);
        array=mxCreateDoubleMatrix(3,3,mxCOMPLEX);
        pr=mxGetPr(array);
        pi=mxGetPi(array);
        memcpy(pr,real,9*sizeof(double));
        memcpy(pr,img,9*sizeof(double));
        /*get the filename*/
```

```
        buflen=mxGetNumberOfElements(prhs[0])+1;
        filename=mxMalloc(buflen);
        mxGetString(prhs[0],filename,buflen);
        /*create new MAT file*/
        file=matOpen(filename,"w");
        if(file==NULL)
            mexErrMsgTxt("can't create appointed file");
            /*write data to MAT file*/
            matPutVariable(file,"String",string);
            matPutVariable(file,"Scalar",scalar);
            matPutVariable(file,"Array",array);
            /*close MAT file*/
            matClose(file);
            mexPrintf("successfully create MAT file:%s\n",filename);
            /*release the memory*/
            mxDestroyArray(string);
            mxDestroyArray(scalar);
            mxDestroyArray(array);
            mxFree(filename);
            }
```

首先所有 MAT 数据文件应用程序的 C 语言代码中必须包含 mat.h，它定义了 mat 函数的原型，并且包含了 matrix.h 头文件。接下来的代码和普通的 MEX 文件应用程序的代码没什么区别，然后用 matOpen 函数打开一个数据文件，再调用 matPutVariable 函数向 MAT 文件中写入数据。最后关闭 MAT 文件（matClose）并释放内存。编译链接生成 MEX 文件，然后运行以下代码：

```
>> clear all
>> mex mexcreatematfile.c
>> mexcreatematfile('matdemo.mat')
successfully create MAT file:matdemo.mat
>> %查看生成了什么
>> what
MAT-files in the current directory d:\MATLAB\work
matdemo
MEX-files in the current directory d:\MATLAB\work
mexcreatematfile
>> %加载数据文件
>> load matdemo
>> whos
  Name        Size                    Bytes    Class
  Array       3x3                       144     double array (complex)
```

```
   Scalar        1x1                    8    double array
   String        1x20                  40    char array
Grand total is 30 elements using 192 bytes
>> Array
Array =
     0     3     6
     1     4     7
     2     5     8
>> Scalar
Scalar =
        2003
>> String
String =
MATLAB is wonderful!
```

有关 mat 函数的详细说明可以参阅 MATLAB 的 help 文档，由于篇幅限制，不再讨论 Fortran 语言 MAT 文件的应用和集成开发环境下的应用。

15.4　MATLAB 引擎技术的应用

MATLAB 计算引擎应用程序的思想和 MATLAB 的 MEX 文件正好相反，MEX 文件是在 MATLAB 环境下调用 C 或者 Fortran 语言程序的手段，而 MATLAB 计算引擎则是在 C 或者 Fortran 语言环境中调用 MATLAB 函数的方法。引擎函数本身用 C 或 Fortran 编写，在 Windows 平台上，它与 MATLAB 的通信是借助 ActiveX 进行的，它可运用于以下场合：

- MATLAB 在其他语言编制的应用程序中被当作数学库程序调用，充分利用 MATLAB 指令简单、计算可靠的优点。
- MATLAB 在专用系统中被当作计算引擎时用，前台是 C 语言等编写的 GUI 图形用户接口，后台由 MATLAB 执行各种复杂的计算分析，从而大大缩短了用户的开发时间。

本节主要介绍如何调用 MATLAB 引擎函数，如何用 C 语言编写调用 MATLAB 引擎的源程序。至于如何编写 Fortran 语言源程序，可参阅 help 文档。

【例 15.7】简单的 C 语言计算引擎应用示例——simpleeng.c

通过本例的程序，可以了解调用 MATLAB 引擎的源程序的一般结构；引擎函数的用法，这些函数都是以 eng 前缀开始的；MATLAB 环境外数据与引擎函数的配合使用。

- 编写源程序 simpleeng.c

示例代码设置如下：

```
/*necessory header file*/
#include "engine.h"
#include "stdio.h"
#include "conio.h"
#define BUFFERLEN 256
```

```
/*main function*/
void main()
{
    Engine *ep;
    char cmd[BUFFERLEN];
    int i=0;
    int status=0;
    /*open the engine*/
    ep=engOpen(NULL);
    if(ep==(Engine*)NULL){
    printf("error:can't open MATLAB engine\n");
    exit(-1);
    }
    /*execute MATLAB command*/
    engEvalString(ep,"A=zeros(1,10);");
    /*wait for a moment...*/
    printf("\n please check the outcome in the MATLAB command line!\n");
    printf("\n press any key to continue...\n");
    getch();
    /*execute other MATLAB commands*/
for(i=10;i<20;i++){
    /*the command to be executed in MATLAB*/
    sprintf(cmd,"A(%d)=fibonacci(%d);",i-9,i);
    /*execute the command*/
    engEvalString(ep,cmd);
    }
    /*wait for a moment...*/
    printf("\n please check the outcome in the MATLAB command line!\n");
    printf("\n press any key to continue...\n");
    getch();
    /*close the MATLAB engine and release memory*/
status=engClose(ep);
if(status!=0){
    printf("can't close the engine\n");
    exit(-1);
    }
    printf("\n MATLAB engine is successfully used.\n");
    }
```

首先在代码的最开始包含了需要的头文件，在所有的 C 语言引擎应用程序中，都必须包含 engine.h 头文件，它声明了所有 eng 函数的原型，并且包含了 matrix.h 头文件，其他的头

文件根据需要包含进去；然后声明了 Engine 类型的指针，该指针类似于打开文件时的文件指针，相当于计算引擎的接口句柄，有了这个指针就可以在 C 语言中执行 MATLAB 的指令了；最后完成代码后应该关闭计算引擎，通过 engClose 函数完成，它关闭指针释放内存。

- 编译连接源程序

采用类似于上一节编译 MAT 数据文件引用程序的方法来编译计算引擎程序，可以在 MATLAB 的命令窗口下，也可以在命令提示符窗口下进行。和生成 MEX 文件不同的是，生成 MEX 文件时，只要在 MATLAB 中完成编译器的配置工作就可以使用 MEX 指令完成编译，因为在配置编译器的时候需要将编译器的选项文件改名为 mexopts.bat 选项文件，并且拷贝到了系统环境目录下。但是在系统路径下 mexopts.bat 文件不能完成 MATLAB 计算引擎引用程序的编译，必须通过 MEX 命令行指定具体的选项文件才可以。代码设置如下：

```
>> mex -f D:\MATLAB7\bin\win32\mexopts\msvc70engmatopts.bat simpleeng.c
```

- 运行 simpleeng.exe

可以直接双击 simpleeng.exe 文件，也可以在 MATLAB 命令行窗口键入以下指令：

```
>> !simpleeng&
```

这时操作系统启动一个 MATLAB 进程（只包含 MATLAB 的命令行窗口），如图 15-1 所示。

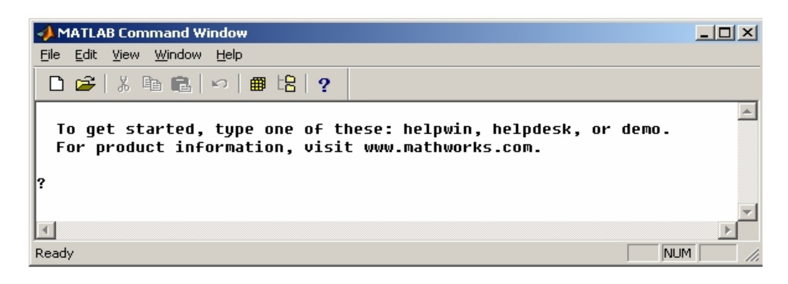


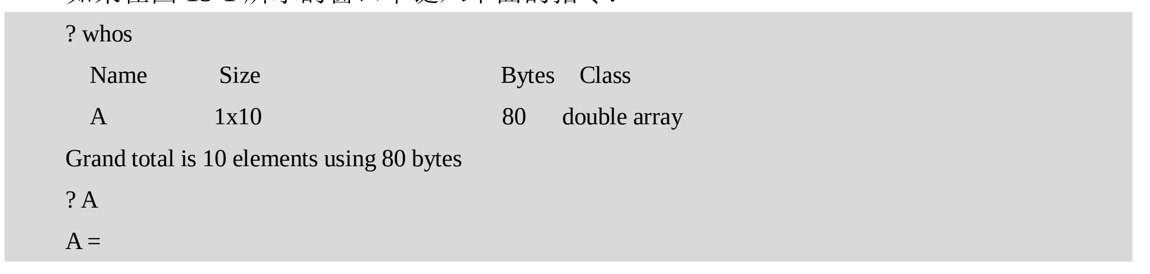
图 15-1　计算引擎启动的 MATLAB 会话窗口

这时在 Windows 的命令行提示符中的显示如图 15-2 所示。



图 15-2　Windows 命令行提示符窗口

如果在图 15-1 所示的窗口中键入下面的指令：



```
? whos
  Name       Size                   Bytes  Class
  A          1x10                   80     double array
Grand total is 10 elements using 80 bytes
? A
A =
```

```
        0       0       0       0       0       0       0       0       0       0
```

在命令提示符行中按任意键继续执行，重复出现提示信息后，回到新启动的 MATLAB 会话窗口，键入下列指令：

```
? A
A=
Columns 1 through 6
        34          35          89          144         233         377
Columns 7 through 10
        610         987         1597        2584
```

可见由于代码的执行 *A* 变成了 Fibonacci 数列的部分元素。然后再按任意键，则执行关闭引擎的命令，MATLAB 会话自动退出运行。由上例可以看出，MATLAB 计算引擎应用程序的基本工作流程如下：

打开计算引擎（engOpen）→设置数据（engPutVariable）→执行 MATLAB 指令（engEvalString）→获取计算结果（engGetVariable）→关闭计算引擎（engClose）。

有关 eng 函数的详细说明可以参看 help 文档。

15.5　MATLAB 的 Java 接口

Java 语言是一种流行的面向对象的高级编程语言，能够完成各种类型的应用程序开发，MATLAB 和 Java 之间的关系是非常密切的，从 5.3 版本开始，MATLAB 中都包含了 Java 虚拟机。在 MATLAB 中可以直接调用 Java 的应用程序，新版本的 MATLAB 工具箱中就包含了很多使用 Java 语言开发的图形用户界面工具，例如 COM Builder 的用户界面 comtool 等。利用 MATLAB 的 Java 接口可以完成下列工作：调用 Java API 类和包，完成核心功能；调用第三方 Java 类；在 MATLAB 环境下创建 Java 对象；通过 Java 语法或者 MATLAB 语法使用 Java 对象的方法；在 Java 对象和 MATLAB 之间交互数据。

Java 可以填补 MATLAB 的一些功能上的空白，特别由于 Java 自身的优势，完全可以通过 Java 语言获取大量的来自互联网或者数据库的数据，而 MATLAB 自身的优势是进行数据分析、科学计算，两者有机结合，充分发挥各自的优势，将极大提高工作效率，节约开发时间和成本。

在 MATLAB 中键入下列指令可以查看当前 MATLAB 使用的 Java 虚拟机的版本：

```
>> version -java
ans =
Java 1.4.2 with Sun Microsystems Inc. Java HotSpot(TM) Client VM
    (mixed mode)
```

可根据此版本选择 JDK（Java 开发包）的版本，避免出现不兼容的现象。

15.5.1　Java 接口应用

Java 是一种面向对象的高级编程语言，其中类和对象是最基本的概念，如果要创建对象，

则必须有相应的类存在。Java 语言除了最基本的关键字以外，还提供了大量的预先编制好的类，应用 Java 就是通过 Java 语言的类完成各种功能。一般可将这些类分为 3 种：Java 内建类（由 Java 语言本身提供的类和类包）；第三方定义类（一些商业化或者免费的应用于专门领域的类包）；用户自定义类（从已有的类派生出来或者直接开发的新类）。

不同的类组合在一起构成类包，这些类包一般安装在系统路径下面，通过系统变量 classpath 定义给操作系统。在 MATALB 中，其系统路径下存在一个文本文件——classpath.txt，该文件定义了 MATLAB 环境可以直接引入的 MATLAB 包。一般该文件位于%MATLABROOT%\toolbox\local 路径下。默认情况下，classpath 文件只是定义了 Java 内建类包和由 Mathworks 公司提供的工具箱类包，如果需要使用第三方或者用户自定义类包，则需在文件中添加，其使用规则是：用户自定义类库添加直到父目录的完整路径，包添加直到最高级父目录的完整路径，JAR 文件中的类库添加完整的路径、完整的文件名。引入 Java 类包后就可直接创建 Java 对象。

在 MATLAB 中创建 Java 对象有直接用 Java 类定义或用 javaObject 函数创建两种方法。下面举例说明。

【例 15.8】在 MATLAB 中键入下列指令

示例代码设置如下：

```
>> %直接创建
>> fa=java.awt.Frame('Frame A');
>> %用 javaObject 方法
>> fb=javaObject('java.awt.Frame','Frame B');
>> whos
  Name          Size                          Bytes   Class
  fa            1x1                                   java.awt.Frame
  fb            1x1                                   java.awt.Frame
Grand total is 2 elements using 0 bytes
```

在上述代码中，都使用了完整的 Java 类的名称，可以在 Java 语言中使用 import 命令导入必要的类包，这样声明对象的时候就不用给出完整的类的名称了，在 M 语言中也是如此，如上例，也可写作下面的代码：

```
>> import java.awt.Frame
>> fa=Frame('Frame A');
```

【例 15.9】设置 Java 对象的属性

示例代码设置如下：

```
>> setTitle(fa,'New A')
>> %获取 fa 的属性
>> title=getTitle(fa)
title =
New A
>> fb.setTitle('New B')
>> title=getTitle(fb)
title =
```

```
New B
```

本例采用了两种方法设置了不同对象的同一个属性。修改 Java 对象的属性还可以像操作句柄图形对象一样使用 set 或者 get 函数。而且在 M 语言中创建的 Java 对象可以通过 save 和 load 命令保存到 MAT 数据文件中和加载到当前的工作空间中。

在应用 Java 对象时需要注意 Java 对象在 MATLAB 中属于一类特殊的数据类型，几乎所有的 MATLAB 数据都可以向 Java 数据对象转变，但是 Java 的对象一般不能直接参与 MATLAB 的数据运算，需要进行相应的类型转换才可以。

【例 15.10】Java 的数据

示例代码设置如下：

```
>> %创建一个双精度的 Java 对象
>> jD=java.lang.Double(33)
jD =
33.0
>> jD+3
??? Function 'plus' is not defined for values of class 'java.lang.Double'.
Error in ==> plus at 14
   builtin('plus', varargin{:});
>> whos
   Name        Size                 Bytes  Class
   jD          1x1                         java.lang.Double
 Grand total is 1 elements using 0 bytes
>> %不能相加，jD 是一种特殊的数据类型
>> %强制类型转换
>> double(jD)
ans =
      33
>> ans+3
ans =
      36
```

最后在 MATLAB 中创建 Java 的数据，需要用到一个函数：

```
javaArray(‘package_name,class_name’,x1,…xn)
```

其中 *x*1～*xn* 是数组每一维的尺寸。

15.5.2 应用示例

在 M 语言中应用 Java 类的主要目的之一就是充分利用 Java 语言的网络功能，例如读取网络上的信息等。

【例 15.11】读取 URL 信息

示例代码设置如下：

```
function readURL(website)
```

```
%read website's data,if no input,read default homepage.
if nargin==0
    website='http://www.sina.com.cn';
end
%create URL object
url=java.net.URL(website);
%create input stream
is=openStream(url);
isr=java.io.InputStreamReader(is);
br=java.io.BufferedReader(isr);
%read network page's data
for i=0:44
    s=readLine(br);
end
readLine(br)
readLine(br)
readLine(br)
```

本例大部分都是 Java 的语法结构，首先创建了 Java 的 URL 类对象，然后利用 Java 的 I/O 流的功能从互联网的网页上读取了信息，这里将互联网的网页看作纯文本文件读取，最后三行每一句都读取文本中的一行。运行结果如下：

```
>> readURL
ans =
A.an02:link {text-decoration:none;color:#1110AC}
ans =
A.an02:visited {text-decoration:none;color:#65038e}
ans =
A.an02:active,A.an02:hover {text-decoration:none;color:#ff0000}
```

15.6　MATLAB 中的 DDE 技术

动态数据交换 DDE（Dynamic Data Exchange）是允许各 Windows 应用程序间交换数据的通信机制。Windows 平台上的 MATLAB 作为一个应用程序，也具有借助 DDE 与其他应用程序通信的功能。

15.6.1　关于 DDE 的一般性说明

应用程序可以借助 DDE 通话实现彼此间的通信。请求建立对话的应用程序称为客户（Client），而响应对话请求的应用程序被称为服务器（Server）。

当客户应用程序创建 DDE 对话时，必须识别被呼叫服务器的两个 DDE 参数：服务名

（Service name），即被请求对话的应用程序名；话题（Topic），即对话主题。这两个参数的组合构成了区分不同对话的惟一标识。

每个作为服务器的应用程序都有惟一的服务名。它们通常就是该应用程序的（不带扩展名）可执行文件名。比如 matlab 是 MATLAB 的服务名。

15.6.2 DDE 中的 MATLAB 服务器

视客户应用程序的具体情况而定，客户可以采用不同方法访问作为服务器的 MATLAB。假如客户应用程序能够提供管理 DDE 对话的函数或宏，则必须充分利用那些函数和宏；假如客户应用程序是自编的，则最好利用 MATLAB 引擎库或直接利用 DDE。MATLAB 用作服务器时的工作原理如图 15-3 所示。此时客户应用程序是通过 DDE 函数与 MATLAB DDE 服务器模块进行通信的。客户 DDE 函数可以由客户应用程序提供，也可以由 MATLAB 引擎库提供。



图 15-3　DDE MATLAB 服务器工作模式

DDE 对话内容由一组预习规定的参数名称组成。当 MATLAB 作为 DDE 服务器使用时，所能选用的具体名称和它们间的层次关系如图 15-4 所示。由图可见，MATLAB 有两类“话题”System 和 Engine。而每类话题又包含几个不同的内容细节。例如，Engine 话题就其内容性质而言又分为两类，即客户把指令发给 MATLAB 计算（EngEvalString）和客户向 MATLAB 索取结果；第二类由于数据性质的不同，又分为文字结果（EngStringResult），图形结果（EngFigureResult），以 Text 格式或 XL 表格式把对话内容约定为索取某个名称的矩阵。

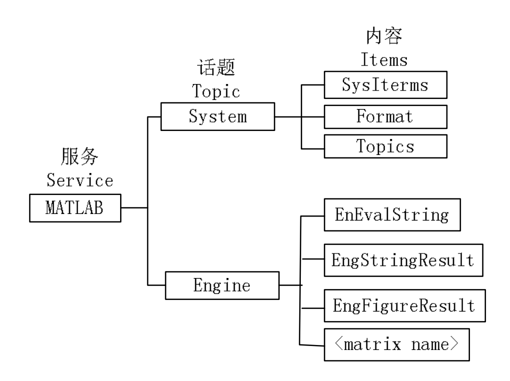


图 15-4　DDE 中 MATLAB 作为服务器使用时定义对话的参数名称层次表

15.6.3 DDE 中的 MATLAB 客户

当 MATLAB 以客户身份建立 DDE 通信时，其工作原理如图 15-5 所示。图中的客户模块可由一系列的函数构成（见附录），本节将通过一个例子展示各指令间的配合使用。



图 15-5　MATLAB 作为客户时的 DDE 原理图

【例 15.12】设计一个 DDE 对话程序。MATLAB 作为客户，Excel 作为服务器，本例演示 DDE 的创建和关闭，热连接的建立和使用。程序（dde_disp.m）如下：

```
clear;
h=surf(peaks(20));   %绘制曲面图形并产生图柄 h
z=get(h,'zdata');       %得到曲面的 Z 坐标数据
chann=ddeinit('excel','Sheet3');   %为两者的 DDE 对话建立通道 chann
range2='r1c1:r20c20';       %为空白表格指定区域，起名为 range2
rc=ddepoke(chann,range2,z);
%借助通道 chann，将数据 z 送到指定位置 range2。操作成功，rc 为 1
%...借助通道 chann，在 MATLAB 与指定区域 range2 之间建立热连接…
rc=ddeadv(chann,range2,'set(h,''zdata'',z);','z');
%Excel 表格 Sheet3 的 range 区域中任何数据的改动，引发如下操作：
%立即引起其后指令的执行；MATLAB 空间中的 z 变量被实时更新。
%...切断 MATLAB 与 range2 的热连接…
hc=uimenu(gcf,'Label','关闭');
hc1=uimenu(hc,'Label','关热连接','Callback','ddeunadv(chann,range2);');
hc2=uimenu(hc,'Label','关闭对话','Callback','ddeterm(chann);');
hc3=uimenu(hc,'Label','关图形窗','Callback','close;');
```

在 Excel 开启的前提下，运行上述程序将产生如图 15-6 和图 15-7 所示的两个界面。

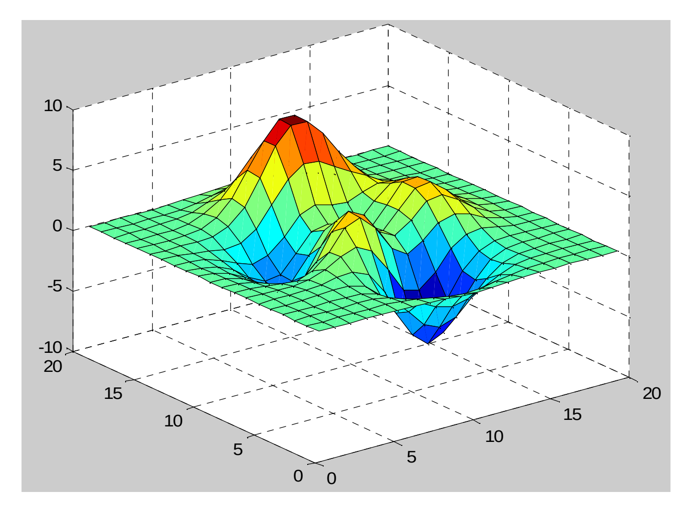
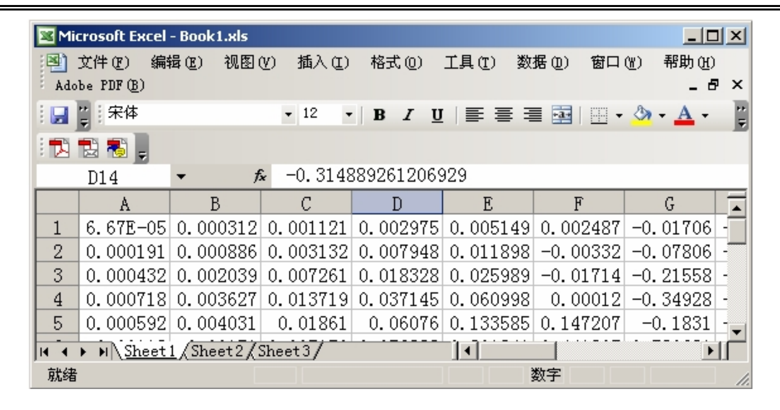


图 15-6　运行后生成的图形

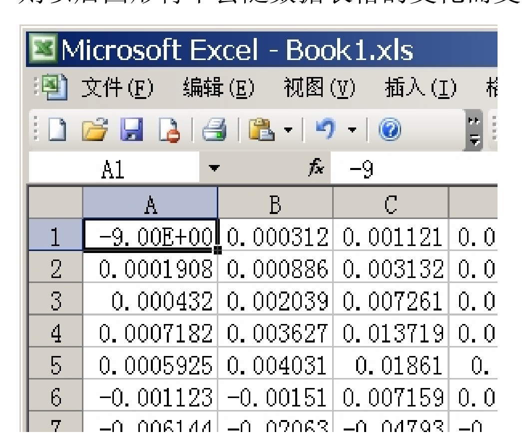


	A	B	C	D	E	F	G
1	6.67E-05	0.000312	0.001121	0.002975	0.005149	0.002487	-0.01706
2	0.000191	0.000886	0.003132	0.007948	0.011898	-0.00332	-0.07806
3	0.000432	0.002039	0.007261	0.018328	0.025989	-0.01714	-0.21558
4	0.000718	0.003627	0.013719	0.037145	0.060998	0.00012	-0.34928
5	0.000592	0.004031	0.01861	0.06076	0.133585	0.147207	-0.1831

图 15-7　运行后产生的表格数据

将 Sheet3 表格中的 *A*1 元素修改为－9，如图 15-8 所示，由于该数据区和所画曲面之间建立了热连接，所以可以观察到图形发生了如图 15-9 的变化，在曲面正前方处产生了一个向下的尖角，这就是热连接的作用使得图形曲面随表格数据的变化同步变化。

所建立的热连接和对话通道若不再使用，则应该关闭节省资源。本例在图形窗上为关闭热连接和对话通道设立了专门的菜单如图 15-10 所示。如果选中其中的“关热连接”选项，则以后图形将不会随数据表格的变化而变化。



	A	B	C
1	-9.00E+00	0.000312	0.001121
2	0.0001908	0.000886	0.003132
3	0.000432	0.002039	0.007261
4	0.0007182	0.003627	0.013719
5	0.0005925	0.004031	0.01861
6	-0.001123	-0.00151	0.007159

图 15-8　改变 A1 的值图

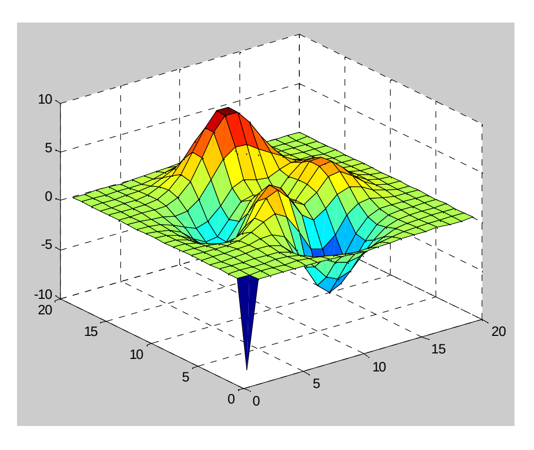


图 15-9　曲面产生相应的下尖角图

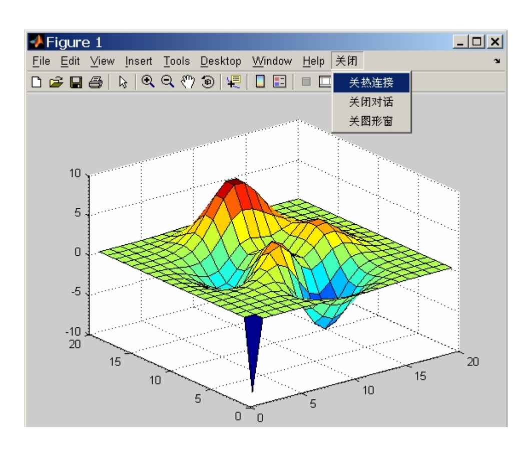


图 15-10　为关闭热连接和对话通道设计的菜单

> 注意：假如建立的通道没有正常关闭，那么同一个程序的再运行可能出现通道识别错误。在这种情况下，改变通道代号或者冷启动计算机可能克服故障。

15.7　MATLAB 中的 ActiveX 技术

15.7.1　关于 ActiveX 的一般性说明

ActiveX 是一种基于 Microsoft Windows 操作系统的组件集成协议，是各种面向对象技术的集合。这些技术共有的基础是“组建对象模型（Component Object Model）”，简称 COM。借助 ActiveX，开发商和终端用户就能把来自不同商家的 ActiveX 组件无缝地集成在自己的应用程序中，从而完成特定的目的。不仅节省了开发时间，而且有效地避免了低水平的重复开发。

每个 ActiveX 都支持一个或多个赋名的界面，而界面是一组逻辑相关方法、属性和事件的组合。方法类似于请求对象实现某动作的函数；属性是对象持有的状态参数；事件是将控制交回客户（Client）的通知，与图形句柄的 callback 相似。MATLAB 支持两种 ActiveX 技术，即 ActiveX 控件封装集成和 ActiveX 自动化。ActiveX 控件是指那些可视、能编程的集成于 ActiveX 容器的应用组件，比如 Microsoft Internet Explorer 和 Web Browser control。

ActiveX 自动化使得 MATLAB 能施控和受控于其他组件。当 MATLAB 受控于其他组件时，表现为自动化服务器（Automation server），其功能包括：在 MATLAB 空间中执行指令，与MATLAB空间直接交换数据；当MATLAB控制其他组件时，表现为自动化客户（Automation client），其功能是 MATLAB 借助 M 文件指示和操纵自动化服务器。

MATLAB 自动化客户的功能仅是 MATLAB ActiveX 控件封装集成功能的子集。所有的 ActiveX 控件都是 ActiveX 自动化服务器，但不是所有的自动化服务器一定是 ActiveX 控件。那些不是控件的自动化服务器将不可被物理地、可视地镶嵌于客户应用中。MATLAB 本身是服务器而不是控件，所以不能被镶嵌在其他客户应用中。然而，由于 MATLAB 是控制容器，所以其他的 ActiveX 控件可以内嵌在 MATLAB 中。

15.7.2　MATLAB 的 ActiveX 自动化

1．MATLAB ActiveX 自动化控制器（客户）

如果要 MATLAB 通过 ActiveX 自动化客户支持调用其他 ActiveX 组件，那么必须先查阅该 ActiveX 组建的相关文件，从中得到该组件的名字（ProgID）、该组件所采用的接口名、方法（Method）、属性（Property）和事件（Event）等。

ActiveX 的创建和操作函数见附录。指令 actcontrol 用于创建 ActiveX 自动化客户支持。该指令运行后将引出指定组件名（比如 Excel、Application）的对象默认界面。通过该对象属性的获取和设置、方法的激活，就可以改变该对象的界面和行为。

【例 15.13】MATLAB 以自动化客户的资格通过 M 脚本文件把 Microsoft Excel 用作自动

化服务器。本例演示调用自动化服务器的基本指令。Excel 默认界面的开启；增添工作簿（Workbook）；改变激活的当前页（Worksheet）；MATLAB 与 Excel 之间的数据传递；Excel 的数据保存。（图 15-11 给出了第 21 行程序执行前产生的 Excel 界面）

程序(actexcel.m)如下：

```
excel=actxserver('Excel.Application');%启动 Excel 并返回名为 excel 的 Activex 服务器对象
disp('为看清 Excel 界面及其变化，请把 MATLAB 界面调整的远小于屏幕！')
disp('按任意键，将可看到“Excel 界面”出现。')
pause
set(excel,'Visible',1);     %使开启的 Excel 默认界面可见
disp('按任意键，可见到 Excel 界面出现第一张表激活的“空白工作簿”。')
pause
wkbs=excel.Workbooks;        %新工作簿句柄
Wbk=invoke(wkbs,'Add');      %产生空白的新工作簿
disp('按任意键，当前激活表由第一张变为指定的第二张。')
pause
Sh=excel.ActiveWorkBook.Sheets;   %当前激活工作簿的表格句柄
sh2=get(Sh,'Item',2);                    %取得第二张表的句柄
invoke(sh2,'Activate');                  %使第二张表为当前激活页
disp('按任意键，把 MATLAB 空间中的 A 矩阵送到 Excel 的指定位置。')
pause
Actsh=excel.Activesheet;                 %当前激活表的句柄
A=[1,2;3,4];
actshrng=get(Actsh,'Range','A1','B2'); %得到当前表指定区域的句柄
set(actshrng,'Value',A);                        %把 A 矩阵送到 Excel 的指定区域
disp('按任意键，获取 Excel 指定区域内的数据，')          %第 21 行
disp('并以 MyExcel.xls 文件形式保存在 D:\MATLAB7\work 目录上。')
pause
rg=get(Actsh,'Range','A1','B2');          %得到 Excel 指定区域句柄
B=rg.value;                                        %获取指定区域上的值
B=reshape([B{:}],size(B));
invoke(Wbk,'SaveAs','D:\MATLAB7\work\MyExcel.xls');%把 Wbk 工作簿保存在指定目录下
disp('按任意键，关闭 excel 句柄代表的 Excel。')
pause
invoke(excel,'Quit');                          %关闭 Excel
```

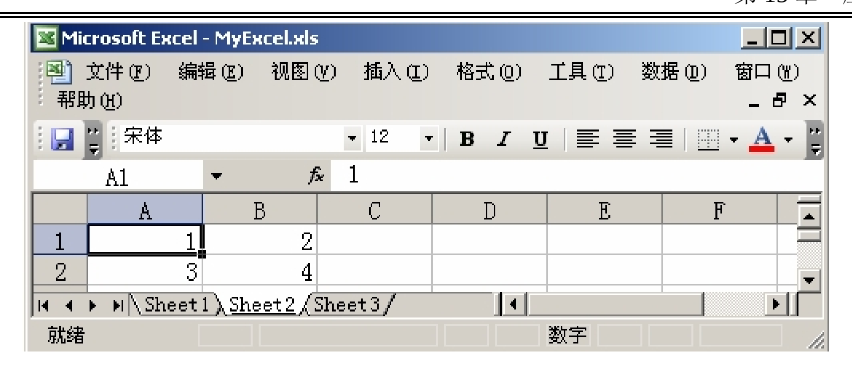


图 15-11　第 21 行程序执行前产生的 Excel 界面

2．MATLAB ActiveX 自动化服务器

通过 MATLAB ActiveX 自动化服务器，用户可以在自己的应用程序中执行 MATLAB 命令，并可以与 MATLAB 的工作空间交换数据。将 MATLAB 作为服务器使用时，用户必须首先查阅希望使用自动化服务器的应用程序的文档。查明如何在控制器中开启自动化服务器；MATLAB ActiveX 对象在系统注册表中定义的名字，即 ProgID。通常 ProgID 取以下两个名字中的一个：MATLAB.Application（将启动的 MATLAB 自动化服务器作为“共享”服务器）；MATLAB.Application.Single（将启动的 MATLAB 自动化服务器作为“专用”服务器独享）。

附　　录

A.1　常用命令和函数

表 1　　特殊变量名表

变量名称	变量含义	变量名称	变量含义
ans	MATLAB 中默认变量	i(j)	复数中的虚数单位
pi	圆周率	nargin	所用函数的输入变量数目
eps	计算机中的最小数，PC 机上为 2^{-52}	nargout	所用函数的输出变量数目
inf	无穷大，如 1/0	realmin	最小可用正实数
NaN	不定值，如 0/0，∞/∞，0*∞	realmax	最大可用正实数

表 2　　运算符与操作符

函数分类	函数名	说明	函数分类	函数名	说明
运算符	＋	加法	关系操作函数	eq(A,B)	等于
	－	减法		ne(A,B)	不等于
	*	矩阵乘法		lt(A,B)	小于
	.*	数组乘法		gt(A,B)	大于
	^	矩阵乘方		le(A,B)	小于等于
	.^	数组乘方		ge(A,B)	大于等于
	\	矩阵右除	逻辑运算符	&	逻辑与
	/	矩阵左除		|	逻辑或
	.\	数组左除		~	逻辑非
	./	数组右除	逻辑运算函数	and(A,B)	逻辑与
关系操作符	= =	等于		or(A,B)	逻辑或
	~ =	不等于		not(A,B)	逻辑非
	<	小于		xor(A,B)	逻辑异或
	>	大于		any(A)	向量 A 中有非零元素时返回 1；矩阵 A 某一列有非 0 元素时此列返回 1
	<=	小于等于		all(A)	向量 A 中所有元素非 0 时返回 1；矩阵 A 中某一列所有元素非 0 时，此列返回 1
	>=	大于等于			

表 3 基本数学函数

函数分类	函数名	说明	函数分类	函数名	说明
三角函数	sin	正弦函数	三角函数	asech	反双曲正割函数
	sinh	双曲正弦函数		cot	余切函数
	asin	反正弦函数		coth	双曲余切函数
	asinh	反双曲正弦函数		acot	反余切函数
	cos	余弦函数		acoth	反双曲余切函数
	cosh	双曲余弦函数	其他常用计算函数	fix	向零方向取整
	acos	反余弦函数		round	四舍五入到最近的整数
	acosh	反双曲余弦函数		floor	向无穷大方向取整
	tan	正切函数		rem	求两整数的相除的余数
	tanh	双曲正切函数		exp	指数函数
	atan	反正切函数		log	自然对数函数
	atanh	反双曲正切函数		log10	以 10 为底的对数函数
	sec	正割函数		sort	开方函数
	sech	双曲正割函数		abs	绝对值函数
	asec	反正割函数			

表 4 测试函数

函数名	说明
isempty(A)	若参量 A 为空，返回 1；否则，返回 0
isglobal(A)	若参量 A 为全局变量，返回 1；否则，返回 0
ishold	当前绘图状态保持是 ON，返回 1；否则，返回 0
isieee	计算机执行 IEEE 运算，返回 1；否则，返回 0
isinf(A)	若参量 A 为无穷大，返回 1；否则，返回 0
isletter(A)	若参量 A 为字母，返回 1；否则，返回 0
isnan(A)	若参量 A 为不定值，返回 1；否则，返回 0
isreal(A)	若参量 A 无虚部，返回 1；否则，返回 0
isspace(A)	若参量 A 为空格字符，返回 1；否则，返回 0
isstr(A)或 ischar(A)	若参量 A 为一个字符串，返回 1；否则，返回 0
isstudent(A)	若 MATLAB 为学生版，返回 1；否则，返回 0
isunix(A)	若计算机为 UNIX 系统，返回 1；否则，返回 0
isvms(A)	若计算机为 VMS 系统，返回 1；否则，返回 0

表 5 矩阵函数

函数分类	函数名	说明
常用矩阵函数	cond	矩阵的条件数值
	condest	1-范数矩阵条件数值
	cross	矩阵的叉积
	det	矩阵的行列式值
	dot	矩阵的点积
	eig	矩阵的特征值
	eigs	矩阵的特征值
	inv	矩阵的逆
	norm	矩阵的范数值
	normest	矩阵的 2-范数值
	rank	矩阵的秩

续表

函数分类	函数名	说明
常用矩阵函数	orth	矩阵的正交化运算
	rcond	矩阵的逆条件数值
	trace	矩阵的迹
	triu	上三角变换
	tril	下三角变换
	diag	对角变换
	exmp	矩阵的指数运算
	exmp1	Pade 法进行矩阵的指数运算
	exmp2	Taylor 级数进行矩阵的指数运算
	logm	矩阵的对数运算
	sqrtm	矩阵的开方运算
	cdf2rdf	复数对角矩阵转换成实数块对角形矩阵
	rref	转换成逐行递减的阶梯矩阵
	rsf2csf	实数块对角形矩阵转换成复数对角矩阵
	rot90	矩阵逆时针方向旋转
	fliplr	左、右翻转矩阵
	flipud	上、下翻转矩阵
	reshape	改变矩阵的维数
	funm	一般的矩阵函数
矩阵分解函数	chol	矩阵的 Cholesky 分解
	eig	矩阵的特征值分解
	hess	矩阵的 Hessenberg 分解
	lu	矩阵的 LU 分解
	null	奇异值分解求得的矩阵的零空间的标准正交基
	qr	矩阵的 QR 分解
	qz	矩阵广义特征值的 QZ 分解
	schur	矩阵的 Schur 分解
	svd	矩阵的奇异值分解
	svds	矩阵的奇异值分解

表 6　稀疏矩阵函数

函数名	说明	函数名	说明
sparse	最常用的稀疏矩阵生成函数	spones	用 1 代替非零元素
spidiags	以对角带生成稀疏矩阵	sprank	稀疏矩阵的秩
spconvert	由外部数据输入生成稀疏矩阵	spy	显示稀疏矩阵的结构图
find	求非零元素在稀疏矩阵中的索引	issparse	判断是否是稀疏矩阵
speye	生成稀疏单位矩阵	colmmd	列最小度
sprand	生成稀疏的均匀分布随机矩阵	colperm	按列排列向量
sprandn	生成稀疏的正态分布随机矩阵	dmperm	矩阵的 DuITnageMendelsohn 分解
sprandsym	生成稀疏的对称随机矩阵	randperm	随机排列向量
full	把稀疏矩阵转化为满矩阵	symmmd	最小对称度
nnz	稀疏矩阵非零元素的个数	gplot	按图论画图
nonzeros	稀疏矩阵非零元素的值	etree	给出矩阵的消元树
nzmax	存放非零元素所需内存	etreeplot	画消元树图
spalloc	为非零元素分配内存空间	treeplot	画结构树图
spfun	求非零元素的函数值		

表 7 特殊矩阵

矩阵名称	说明
[]	空矩阵
zeros	零矩阵
ones	1 矩阵
eye	单位矩阵
rand	均匀分布的随机矩阵
randn	正态分布的随机矩阵
company	伴随矩阵
magic	魔方矩阵
hilb	Hibert 矩阵
invhilb	逆 Hibert 矩阵

表 8 多项式运算函数

函数名	说明
conv	多项式乘法
deconv	多项式除法
poly	用根生成多项式
polyder	多项式导数
polyfit	多项式拟合
polyval	多项式求值
polyvalm	多项式求值
ppval	分段多项式求值
residue	部分分式展开
roots	多项式求根

表 9 数据分析函数

函数分类	函数名	说明
通用的数据分析函数	max	求最大元素
	min	求最小元素
	mean	求元素平均值或列元素的平均值
	median	求列元素的中值
	std	求元素标准差
	sum	求各列元素的和
	prod	求列元素的积
	hist	直方图
	trapz	梯形法求数值积分
	quad	Simpson 法求数值积分
	quad8	Cotes 法数值积分
	cumprod	求列元素的累计积
	cumsum	求列元素的累计和
	cumtrapz	梯形法求累计数值积分
	sort	按升序对元素进行排序
	sortows	按升序排列矩阵各行
有限差分函数	dff	求差分和近似导数
	gradient	求近似梯度
	del2	求离散 Laplace 算子
相关关系函数	corrcoef	求相关系数
相关关系函数	cov	求协方差矩阵
	subspace	求两个子空间之间的夹角

表 10　　傅立叶变换函数

函数名	说明	函数名	说明
filter	一维离散时间滤波器	ifft2	二维逆快速傅立叶变换
filter2	二维离散时间滤波器	abs	模
conv	卷积	angle	相角
conv2	二维卷积	unwrap	消除相角突变
fft	快速傅立叶变换	fftshift	平移零点到频率中心
fft2	二维快速傅立叶变换	cplxpair	把数字分类为复共轭对
ifft	逆快速傅立叶变换	nexttpow2	最靠近 2 的次幂

表 11　　符号工具箱函数

函数分类	函数名	说明
符号表达式运算	sym	建立符号表达式
	syms	建立符号表达式
	numden	提取分子与分母
	symadd	符号加法
	symsub	符号减法
	symmul	符号乘法
	symdiv	符号除法
	sympow	符号表达式的幂运算
	symop	符号运算
	compose	符号表达式的复合函数运算
	finverse	符号表达式的反函数运算
	symsum	求表达式的符号和
	symvar	求符号变量
	numeric	符号表达式转换为数值表达式
	eval	符号表达式转换为数值表达式
	sym	数值表达式转换为符号表达式
	poly2sym	将等价系数向量转换为它的符号多项式
	sym2poly	将符号多项式转换为它的等价系数向量
	charpoly	特征多项式
符号可变精度运算	digits	设置可变精度
	vpa	可变精度计算
符号表达式的化简	pretty	显示方便查看的符号表达式
	collect	合并符号表达式的同类项
	horner	把一般的符号表达式转换为嵌套形式的符号表达式
	factor	对符号表达式进行因式分解
	expand	对符号表达式进行展开
	simple	对符号表达式进行化简
	simplify	求解符号表达式的最简形式
符号矩阵的运算	transpose	符号矩阵的转置
	determ	符号矩阵的行列式运算
	det	符号矩阵的行列式运算
	inv	符号矩阵求逆运算
符号矩阵的运算	rank	符号矩阵求秩运算
	eig	求符号矩阵的特征值、特征向量
	eigensys	求符号矩阵的特征值、特征向量
	svd	符号矩阵的奇异值运算
	singvals	符号矩阵的奇异值运算
	jordan	符号矩阵的约当标准型运算

续表

函数分类	函数名	说明
符号微积分	limit	符号极限
	diff	符号微分
	int	符号积分
	taylor	泰勒级数展开
符号画图	ezplot	符号函数画图
	fplot	符号函数画图
符号方程的求解	solve	代数方程的求解
	linsolve	齐次线性方程组的求解
	fsolve	非线性方程组的求解
	dsolve	微分方程的求解

表 12　　MATLAB 低级文件 I/O 函数

函数名	说明	函数名	说明
fclose	关闭文件	fscanf	从文件中读格式化数据
feof	检查是否到文件尾	fseek	设置文件指针的位置
ferror	查询文件 I/O 的出错信息	ftell	获取文件指针的位置
fgetl	从文件中读一行，并忽略回车符	fwrite	向一个文件中写入二进制数据
fgets	从文件中读一行，并包括回车符	sprintf	把格式化数据写入字符串
fopen	打开文件	sscanf	从格式化字符串中读入数据
fprintf	把格式化数据写到文件中	tempname	建立临时文件名
fread	从文件中读二进制数据	tempdir	读出临时文件名
frewind	将打开的文件的指针反绕		

表 13　　程序设计与文件调试函数

函数分类	函数名	说明
程序设计标识函数	script	命令式 M 文件
	function	函数式 M 文件
	eval	执行字符串
	feval	执行字符串定义的函数
	global	定义全局变量
程序结构控制流	for	预定的次数重复执行的循环语句
	end	与 for，while，if，switch 语句匹配的结束标志语句
	while	以不定次数执行的循环语句
	if	条件执行语句
	elseif	与 if 匹配使用
	else	与 if 匹配使用
	switch	分支选择语句
	case	与 switch 匹配使用
	otherwise	与 swich 匹配使用
	continue	结束本次循环，判断是否执行下一次循环
	break	终止本次循环，跳出最内层的循环
程序结构控制流	return	返回调用函数
	echo	显示执行的 M 文件的每条命令
	error	出错信息显示
	try	对异常进行处理
	catch	与 try 匹配使用

续表

函数分类	函数名	说明
交互式输入	input	提示用户输入
	keyboard	通过键盘输入数据
	pause	等候用户响应
调试命令	dbclear	取消断点
	dbcont	断点后重新运行
	dbdown	工作空间下移
	dbquit	退出调试模式
	dbstack	列表显示堆栈调用
	dbstatus	列表显示所有的断点
	dbstep	执行一行或多行
	dbstop	设置断点
	dbtype	列表显示带行号的 M 文件
	dbup	工作空间上移

表 14 字符串函数

函数名	说明	函数名	说明
Isstr	判断是否是字符	abs	把字符串变成 ASCII 码值
Blanks	空格符	setstr	把 ASCII 码值变成字符串
deblank	删除空格符	num2str	把数字变成字符串
eval	执行字符串	str2num	把字符串变成数字
isletter	确定字符串中的字符是否为字母	str2mat	把字符串变成文本矩阵
strcmp	字符串比较	hex2num	把十六进制字符串变成 IEEE 浮点数
findstr	在其他字符串中寻找字符串	hex2dec	把十六进制字符串变成十进制数
strrep	用一个字符串代替另一个字符串	dec2hex	把十进制数变成十六进制字符串
upper	把字符串中的字符变成大写形式	sprintf	把带格式的数字变成字符串
lower	把字符串中的字符变成小写形式	sscanf	把字符串变成带格式的数字

表 15 二维图形函数

函数分类	函数名	说明
基本二维绘图函数	plot	二维线形图
	loglog	双对数坐标
	semilogx	单对数坐标图，x 轴为对数坐标，y 轴为线性坐标
	semilogy	单对数坐标图，y 轴为对数坐标，x 轴为线性坐标
	polar	极坐标图
	plotyy	双（y）轴图
坐标轴控制函数	axis	控制坐标轴比例和外观
	zoom	放大或缩小
	grid	网格显示控制
	box	坐标轴外框显示
	hold	控制是否保持当前图形
	axes	在指定位置创建坐标轴
	subplot	创建子图图区
硬拷贝和打印函数	print	打印图形或仿真系统，或保存图形到 M 文件
	printopt	打印机默认设置
	orient	设置纸张方向

续表

函数分类	函数名	说明
图形标注函数	Plotedit	编辑和标注图形工具
	legend	图形图例
	title	图形标题
	xlabel	x 轴标签
	ylabel	y 轴标签
	textlabel	文本标签
	text	文本注解
	gtext	用鼠标放置文本

表 16 三维图形函数

函数分类	函数名	说明
基本三维图形函数	plot3	三维线形图
	mesh	网格图
	surf	表面图
	fill3	填充的三维多边形
颜色控制函数	colormap	颜色查看表
	caxis	颜色按坐标轴比例设置
	shading	颜色阴影模式
	hidden	网格隐藏
	brighten	加亮或变暗色图
	colordef	颜色默认设置
	graymon	灰度监视器图形默认设置
光照控制函数	surf1	给三维阴影表面添加光照
	lighting	光照模式
	material	材料反射模式
	specular	特殊反射
	diffuse	漫反射
	surfnorm	表面法向
相机控制函数	campos	相机位置
	camtarget	相机目标
	camva	相机视角
	camup	相机抬升向量
	camproj	相机投影
透明控制函数	alpha	透明模式
	alphamap	透明查看表
	alim	透明比例
高水平相机控制函数	camlight	创建和设置光线的位置
	lightangle	光线的极坐标位置
视点控制函数	view	指定三维图的视点
	viewmtx	察看转换矩阵
	rotate3d	交互旋转三维图

表 17 特殊图形函数

函数分类	函数名	说明
特殊二维图形函数	area	面积图
	bar	二维条形图
	barh	二维水平条形图
	comet	彗星图
	errorbar	误差条图
	ezplot	简易函数绘图器
	ezpolar	简易极坐标绘图器
	feather	羽列图
	fill	填充的二维多边形
	fplot	函数图形
	hist	直方图
	pareto	帕雷托图
	pie	饼图
	plotmatrix	散点图矩阵
	scatter	散点图
	stem	火柴杆图
	stairs	阶梯图
等值线图函数	contour	等值线图
	contourf	填充的等值线图
	contour3	三维等值线图
	clabel	给等值线图添加标签
	ezcontour	简易等值线图生成器
	ezcontourf	简易填充等值线图生成器
特殊三维图参数	bar3	三维条形图
	bar3h	三维水平条形图
	comet3	三维彗星图
	ezgraph3	简易一般表面绘图器
	ezmesh	简易三维网格绘图器
	ezmeshc	简易三维网格图叠加等值线图生成器
	ezplot3	简易三维参数曲线绘图器
	ezsurf	简易曲面绘图器
	ezsurfc	简易三维曲面叠加等值线图绘图器
	meshc	网格图叠加等值线图
	meshz	窗帘图
	pie3	三维饼图
	ribbon	带形图
	scatter3	三维散点图
	stem3	三维火柴杆图
	surfc	曲面图叠加等值线图
	trisurf	三角形表面图
	trimesh	三角形网格图
	waterfall	瀑布图
实体模型函数	cylinder	创建柱体
	sphere	创建球体
	ellipsoid	创建椭球体
	patch	创建阴影
	surf2patch	将表面数据转换为阴影数据

续表

函数分类	函数名	说明
动画和快照函数	capture	抓取当前图形
	moviein	初始化动画框内存
	getframe	获取动画框
	movie	演示记录的动画框
	rotate	旋转指定了原点和方向的对象
	frame2im	将动画框转换为索引图片
	im2frame	将索引图片转换为动画框
体积和向量可视化函数	vissuite	可视组合
	isosurface	等表面提取器
	isonormals	等表面法向
	isocaps	等表面终端帽盖
	isocolors	等表面和阴影颜色
	contourslice	切片面板中的等值线
	slice	体积切片图
	streamline	二维、三维向量数据的流线图
	stream3	三维流线图
	stream2	二维流线图
	quiver3	三维矢量图
	quiver	矢量图
	divergence	向量场的差异
	curl	向量场的卷曲和角速率
	coneplot	三维流锥图
	streamtube	三维流管图
	streamribbon	流带图
	streamslice	切片面板中叠加流线图
	streamparticles	流沙图
	interpstreamspeed	内插源于速度的流线顶点
	reducevolume	减少体积数据
	volumebounds	为体积数据返回 x，y，z 和颜色限制
	smooths	平滑三维数据
	reducepatch	减小阴影表面的个数
	shrinkfaces	减少阴影表面的大小
图形显示和文件 I/O 函数	image	显示图片
	imagesc	数据比例化并图形显示
	colormap	颜色查看表
	gray	线性灰度色图
	contrast	灰度色图，用于改进图片的对比度
	brighten	加亮或变暗色图
	colorbar	显示色图
	imread	从图形文件中读取图片
	imwrite	从图形文件中写图片
	imfinfo	图形文件的信息
与颜色相关的函数	spinmap	旋转色图
	rgbplot	绘色图
	colstyle	用字符串说明颜色和风格
	ind2rgb	将索引图片转换为 RGB 图片

表 18 图形窗口的创建和控制函数

函数名	说明
figure	创建图形窗口
gcf	获取当前图形的图柄
clf	清除当前图形
shg	显示图形窗口
close	关闭图形
refresh	刷新图形
openfig	打开已保存图形的新拷贝或显示已存在的拷贝

表 19 插值与拟合

函数名	说明
interp1	一维线性插值
interp2	二维线性插值
interp3	三维线性插值
interft	快速 Fourier 变换得到一维插值
intern	多数线性插值
spline	三次样条插值
polyfit	最小二乘法拟合

表 20 非线性数值解法

函数名	说明
ode23	二三阶 Runge-Kutta 求解常微分方程
ode23p	二三阶 Runge-Kutta 求解常微分方程并画出结果图
ode45	四五阶 Runge-Kutta 求解常微分方程
fmin	求一元函数的极小值
fmins	求多元函数的极小值
fzero	求一元函数的零点值

表 21 GUI 图形函数

函数名	说明
set	设置对象属性
get	获取对象属性
reset	把对象属性重设为默认值
delete	删除对象
gcf	获取当前图形对象的句柄
gca	获取当前图形窗口内当前坐标轴的句柄
gco	获得当前图形窗口内当前对象的句柄
gcbo	获取当前回调对象的句柄
gcbf	获取当前回调图形的句柄
findobj	获取指定的属性值的对象的句柄
drawnow	绘图
copyobj	图形对象拷贝
isappdata	核对应用程序定义的数据是否存在
getappdata	获取应用程序定义的数据的值
setappdata	设置应用程序定义的数据的值
rmappdata	删除应用程序定义的数据
figure	创建图形对象
axes	创建坐标轴对象

续表

函数名	说明
line	创建线条对象
rectangle	创建矩形
light	创建光线
text	创建文本对象
patch	创建补片对象
surface	创建曲面对象
image	创建图像对象
uimenu	创建下拉式菜单对象
uicontextmenu	创建内容式菜单对象
uicontrol	创建控件对象
dialog	创建对话框
uigetfile	创建文件打开对话框
uiputfile	创建文件保存对话框
uisetcolor	创建颜色设置对话框
uisetfont	创建字体设置对话框
pagesetupdlg	创建打印页面设置对话框
pagedlg	创建打印页面设置对话框
printpreview	创建打印预览对话框
printdlg	创建打印对话框
helpdlg	创建帮助对话框
errordlg	创建出错信息提示对话框
msgbox	创建信息提示对话框
questdlg	创建问题显示对话框
warndlg	创建警告信息显示对话框
inputdlg	创建变量输入对话框
listdlg	创建列表选择对话框

A.2　SIMULINK 的库模块

A.2.1　库模块

Demos library　　演示子库
Simulink　　SIMULINK 基本库

A.2.2　连续模块子库 Continuous

Continuous　　连续模块子库
Derivative　　求导数模块
Integrator　　连续函数积分
Memory　　记忆模块
State-Space　　状态方程模块
Transfer Fcn　　传递函数模块

A.2.3 离散模块子库 Discrete

Discrete	离散模块子库
Discrete Filter	离散滤波器模块
Discrete-Time Integrator	离散时间积分模块
Discrete Transfer Fcn	离散传递函数模块
Discrete Zero-Pole	离散零极点增益模块
Unit Delay	单位延迟模块
Zero-Order Hold	零阶保持模块

A.2.4 解析函数和查表函数模块子库 Functions&Tables

Fcn	C 语言格式的任意函数模块
Functions&Tables	解析函数和查表函数模块子库
MATLAB Fcn	MATLAB 语言格式的任意函数
Look-Up Table	一维查表函数模块
Look-Up Table(2-D)	二维查表函数模块

A.2.5 一般数学函数子库 Math

Abs	取绝对值模块
Combinatorial Logic	组合逻辑模块
Gain	增益模块
Logical	逻辑运算模块
MinMax	取极大值或极小值的模块
Math	一般数学函数子库
Mux	复用模块
Product	乘法器
Relational	关系运算模块
Sign	符号运算模块
Slider	滑键增益模块
Sum	求和模块

A.2.6 非线性模块子库 Nonlinear

Dead Zone	死区非线性模块
Nonlinear	非线性模块子库
Relay	继电器模块子库
Saturation	饱和非线性模块

A.2.7　信号和系统模块子库 Signal&Systems

Demux	分用模块
Enable	使能模块
Ground	接地模块
In1	输入端口模块
Merge	汇合模块
Out1	输出端口模块
Signal&Systems	信号与系统模块子库
SubSystem	子系统模块
Trigger	触发模块
Terminator	终端模块

A.2.8　信宿模块子库 Sinks

Display	数值显示模块
Scope	示波模块
Sinks	信宿模块子库
Stop	终止仿真
To File	把数据存为文件
To Workspace	把数据写为矩阵变量
XY Graph	显示 X-Y 图形

A.2.9　信源模块子库 Sources

Clock	连续仿真时钟模块
Constant	恒值输出模块
From File	从文件读数据
From Workspace	从内存读数据
Pulse	脉冲发生器
Signal Generator	信号发生器
Sine Wave	正弦波输出
Sources	信源模块子库
Step	阶跃输出

A.3　应用程序接口函数库

A.3.1　外部程序接口函数库

mexCallMatlab	调用 MATLAB 内建函数、M 文件等
mexErrMsgTxt	发布出错信息并中断 MEX 执行
mexEvalString	利用 MATLAB 计算送去指令
mexFunction	C 语言 MEX 文件的固定格式的接口函数
mexGetVariable	将其他空间内的变量送入 MEX 空间
mexPutVariable	将 MEX 空间内的变量送入其他空间
mexWarnMsgTxt	仅发布出错信息
mxCalloc	配置动态内存
mxSetClassName	为变量命名
mxCreateDoubleMatrix	创建二维未赋值双精度浮点数组
mxCreateNumericMatrix	创建数值矩阵
mxCreateString	创建字符串
mxDestroyArray	释放动态分配的内存
mxGetDimensions	获取指定（高维）数组内各维的大小
mxGetM	获取指定数组的行数
mxGetN	获取指定数组的列数
mxGetNumberOfDimensions	获取指定数组的维数
mxGetNumberOfElements	获取数组所含元素总数
mxGetPi	获取指定数组的虚数部分的数据指针
mxGetPr	获取指定数组的实数部分的数据指针
mxGetString	获得字符串
mxIsChar	判断指定数组是否字符串类型
mxIsComplex	判断指定数组是否为复数型
mxIsDouble	判断指定数组是否为双精度类型
mxIsFromGlobalWS	判断数组来自 MATLAB 全局空间与否
mxIsNumericMatrix	判断指定数组是否为数值类型
mxSetM	设置数组的行数
mxSetN	设置数组的列数
mxSetName	设置数组名

A.3.2　MAT 文件库函数

matClose	关闭 MAT 文件
matDeleteArray	从 MAT 文件中删除一个数组
matPutVariable	向 MAT 文件中写入 mxArray 变量
matGetVariable	从 MAT 文件中读取 mxArray 变量

matOpen 开启 MAT 文件
matPutArrayb 将数组写入到 MAT 文件中
matPutVariableAsGlobal 将数组以全局变量形式写入 MAT 文件

A.3.3 MATLAB 引擎函数库

engClose 关闭 MATLAB 引擎
engEvalString 调用引擎计算字符串表达式
engGetVariable 从 MATLAB 引擎空间中获取变量
engOpen 启动 MATLAB 引擎
engPutVariable 将 mxArray 变量送入 MATLAB 引擎空间
engOutputBuffer 产生保存 MATLAB 输出的缓冲区
MessageBox 在根屏幕上建立对话框

A.3.4 ActiveX 对象的构造和操作命令

actxcontrol 建立一个 ActiveX 控件对象
actxserver 建立一个 ActiveX 自动化服务器对象
delete 删除一个 ActiveX 对象
get 从接口获取属性值
invoke 激活窗口
move 在父窗上移动对象或改变其大小
propedit 要求控件显示其内建的属性页
release 释放 ActiveX 对象
send 显示事件列表
set 对接口属性进行设置

A.3.5 动态数据交换函数

ddeinit 创建 DDE 对话通道
ddeexec 向服务器发送需执行的字符串
ddepoke 向服务器发送数据
ddereq 从服务器得到数据
ddeterm 关闭 DDE 对话通道
ddeadv 建立热连接
ddeunadv 释放热连接
Engine MATLAB 作 DDE 服务器的引擎工作模式
EngEvalString 向 MATLAB 发送需执行的字符串
EngFigureResult 从 MATLAB 获取图形结果
EngStringResult 从 MATLAB 获取非图形结果